NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

A DECISION SUPPORT SYSTEM FOR
COST-EFFECTIVENESS ANALYSIS FOR CONTROL AND
SECURITY OF COMPUTER SYSTEMS

by

Emmanuel A. Prevenas

September 1985

Thesis Advisor: Tung X. Bui

Approved for public release; distribution is unlimited

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE *(and Subtitle)* A Decision Support System for Cost-Effectiveness Analysis for Control and Security of Computer Systems		5. TYPE OF REPORT & PERIOD COVERED Master's Thesis September 1985 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Emmanuel A. Prevenas		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, CA 93943-5100		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, CA 93943-5100		12. REPORT DATE September 1985 13. NUMBER OF PAGES 151
14. MONITORING AGENCY NAME & ADDRESS*(if different from Controlling Office)*		15. SECURITY CLASS. *(of this report)* UNCLASSIFIED 15a. DECLASSIFICATION/ DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Approved for public release; distribution is unlimited

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

decision support system, cost-effectiveness, exposure, control, control set

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

The increasing number of computer failures and crimes has forced managers to tighten the control procedures of their EDP systems. However the cost of an exhaustive control strategy is often very expensive, and its effectiveness is not guaranteed. This study designs and implements a Decision Support System that helps determine optimal control procedures for EDP systems (CEA-DSS). The model base of the proposed DSS consists of various (Continued)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

S/N 0102-LF-014-6601

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

ABSTRACT (Continued)

techniques for estimating computer exposures. The latter can be interactively analyzed via a Dialogue interface that supports tabular and graphic outputs. CEA-DSS also provides extensive database management capabilities to keep track of the diverse control problems. It is implemented in Pascal for the IBM-PC.

Approved for public release; distribution is unlimited.

A Decision Support System for Cost-Effectiveness Analysis for Control and Security of Computer Systems

by

Emmanuel A. Prevenas
Lieutenant, Hellenic Navy
B.S., Naval Academy of Greece, 1973

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN INFORMATION SYSTEMS

from the

NAVAL POSTGRADUATE SCHOOL
September 1985

ABSTRACT

The increasing number of computer failures and crimes has forced managers to tighten the control procedures of their EDP systems. However the cost of an exhaustive control strategy is often very expensive, and its effectiveness is not guaranteed. This study designs and implements a Decision Support System that helps determine optimal control procedures for EDP systems (CEA-DSS).

The model base of the proposed DSS consists of various techniques for estimating computer exposures. The latter can be interactively analyzed via a Dialogue interface that supports tabular and graphic outputs. CEA-DSS also provides extensive database management capabilities to keep track of the diverse control problems. It is implemented in Pascal for the IBM-PC.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENTS

At the completion of this research the author wishes to express his gratitude as well as his personal sincere appreciation to professors T. X. Bui and N. R. Lyons for their assistance.

Furthermore he would like to express his sincere appreciation to his parents Antonios and Alexandra for their spiritual support.

Finally, the author dedicates this thesis to his wife Anastasia, who has always encouraged and helped him during his efforts for education and continuous self improvement.

I. INTRODUCTION

A. DEFINITION OF THE PROBLEM

Management's concern over adequate controls is useless if the data processing system designers, EDP auditors and their managers, do not have the proper training and control techniques to utilize when designing or reviewing the controls associated with computer systems.

No one has ever made a convincing estimate of the total cost of intentional and unintentional loss-causing acts associated with Electronic Data Processing (EDP) processes, but it is clear that the cost is high. Recently, many articles in professional journals as well as textbooks on EDP controls have been published responding to the urgency of protection and prevention of computer failures and frauds. Most of these studies focus on the identification of potential exposures, understanding of current control technology and the elaboration of EDP audit trails. These articles also refer to the importance of estimating costs and benefits, the integration of different audit processes, and the various natures of computer failures and corresponding protection and prevention measures [Ref. 1 and 2]. However a more formalized methodology remains to be desired.

As a consequence of this lack of formalized framework, the design of EDP control systems frequently relies on subjective estimations of the 'EDP controller' or the 'evaluator' for performing Cost-Effectiveness Analysis (CEA). This approach has two major disadvantages. First, the dense and complex inter-relationships between potential computer errors and related types of control procedures may make difficult, if not impossible, for the EDP auditor to capture the totality of the problem. Second, the combined

use of control procedures may cause uncontrollable and undesirable effects. For example, over-auditing reduces the throughput of the computer system due to delays caused by redundant control measures, or under-auditing reduces the protection effectiveness due to incomplete control measures.

B. THE NEED FOR CONTROL AND SECURITY OF COMPUTER SYSTEMS

The management of an entity is responsible for establishing and maintaining adequate controls. The establishment and maintenance of a system of controls is a significant management obligation.

A complex on-line data communication-oriented system consists of various combinations of hardware, software, facilities, people, and the policies and procedures that interrelate these components. The many diverse components and potential entry-points into a complex on-line system make it possible for a person, with sufficient technical or applications knowledge, to enter the system and make unauthorized manipulations of data, programs, or operational procedures. Furthermore, control procedures for an on-line system cut across many lines of responsibility within an organization, creating a control problem in itself.

As the number of more sophisticated computer installations increases rapidly, computers are taking on increasingly responsible work. The more vital the work of the computer, the more important is to protect it from failure and catastrophe, and from criminals and people who misuse its power. The following are typical cases of critical computer implementations [Ref. 3]:

- A large city uses a computer for controlling its police operations. All police vehicles and ambulances are dispatched by men using terminals that inform them of the current emergencies. If the computer system was put out of action, many of the operations could not be controlled.
- 747s approaching a congested airport are prevented from colliding by a computerized air traffic control

system. The air traffic density has been allowed to increase to such a level that it could not be handled without the computer system.

- A variety of nuclear weapon systems are under computer control. The decision to launch a defensive nuclear attack is made by men reacting quickly to information from computer systems.
- Commercial data banks contain trade secrets and other information that could be worth many millions of dollars to the competitors.

Functions like these demand for data integrity, security and privacy. The data processing function must not loose vital data, introduce errors into them and permit unauthorized persons to read or modify the data.

C. SCOPE OF THE THESIS

A conventional life cycle of a computer audit process consists of the following six phases:

1. Information gathering.
2. Evaluation of current control technique.
3. Identification of new control measures or strategies.
4. Selection of control strategy.
5. Implementation.
6. Ex-post evaluation.

This thesis concentrates only on the fourth phase, the selection of control strategy, attempting to apply the Decision Support Systems (DSS) technology into the cost effectiveness auditing process.

D. OBJECTIVE

The objective of the thesis is to introduce a DSS for CEA. This may help EDP auditors and computer center managers to design successful EDP control and security systems, and monitor the effectiveness of the existing ones.

The issue of interactiveness seems to be critical in this context since the process of controlling EDP systems is

expected to be not frequent. The importance of interactiveness is further accentuated when EDP controllers face a large combination of controls. Assuming that the DSS learning curve of the end-user is low to none, the proposed DSS emphasizes on the user friendliness of the system.

E. CHAPTER OUTLINE

Chapter 2 gives a summary description of the CEA Model that the DSS attempts to apply. The third chapter provides a framework addressing user requirements and functions that the DSS has to meet.

The fourth chapter is concerned about the detail design of the Dialog Component of the system. The fifth chapter discusses the design of the Model Component. The sixth chapter describes the design of the Data Component, and the seventh chapter focuses on the Database design which is part of the Data Component.

The implementation of the DSS, along with implementation problems encountered, is discussed in chapter 8. Chapter 9 gives an example of the system's operation simulating the selection of control strategy process.

Finally, possible future extensions of the proposed DSS and concluding comments are discussed in the last chapter.

II. THE CEA MODEL

The purpose of a cost-effectiveness analysis is to determine the most cost effective control strategy to reduce or eliminate potential errors and failures. It has been a generally accepted view that CEA is best used when it is integrated in the whole audit process. Some definitions of the basic concepts are necessary to the understanding of the CEA Model [Ref. 4].

A. DEFINITIONS OF BASIC CONCEPTS

1. The Concept of Exposures

The key element to start a CEA is not control but exposure. The concept of exposure is based on the assumption that the degree of vulnerability of computer systems may be reduced by enforcing EDP control measures, but cannot be totally eliminated due to some errors that remain unpredictable or unable to fully corrected.

2. Costs of Controls

Costs of EDP controls include all costs associated with the design, implementation and use of the controls. With experience gained in designing and implementing control systems, the costs become easier to be identified and quantified.

3. Benefits of Controls

The identification and quantification of benefits derived from control measures is very difficult. One way to look at benefits is to interpret them as a function control effectiveness.

4. Effectiveness of Controls

The effectiveness of a control is the extent to which this control can reduce or minimize the probability that an exposure occurs, reduce the damage if an exposure happens, and/or recover quickly from a damage. Therefore the reliability or performance of a control can be expressed as a percentage of control effectiveness relative to the related exposure.

5. Interdependencies between Controls

Often, a control, though primarily aimed at correcting a specific exposure, may affect one or more other exposures. Such interdependencies may dramatically affect the effectiveness of an EDP control system.

B. ASSUMPTIONS

The model assumes that the following conditions hold:

- Managers and auditors have limited time and capital resources for EDP controls.
- Each corporate computer system is characterized by its specific and unique control structure.
- Independence between potential failures or errors within a computer system.
- Each applied control is expected to prevent, correct or eliminate one or more potential errors, and/or affect others positively or negatively.
- Costs for EDP controls are known and quantifiable .

C. SUMMARY DESCRIPTION OF THE MODEL

Table 1 lists all the variables involved in the mathematic formulas of the model. The CEA Model consists of the following steps:

1. Define all Possible Control Sets

A control set is simply a combination of different available EDP controls. If there are n independent controls,

TABLE 1

DEFINITION OF VARIABLES USED BY THE MODEL

Symbol	Description
m	Number of potential errors or exposures
n	Number of individual control activities
a_i	Control activity, where i = 1 to n
c_i	Costs of implementing a_i
S	Number of control sets
s_k	Control set, where k = 1 to S
e_j	Potential error or exposure, where j = 1 to m
$Pr(e_j)$	Probability that e_j occurs
d_j	Amount of damage when e_j occurs
l_j	Expected damage caused by e_j
f_{ij}	Effectiveness of control a_i on exposure e_j
v_i	Expected benefits obtained from a_i
V_k	Expected benefits obtained from s_k
L_k	Expected loss resulted in using s_k
C_k	Costs of implementing s_k
TC_k	Total cost associated with s_k

the maximum number of control sets is defined as follows:

$$S = \sum_{i=1}^{n} [n! / (i! \cdot (n - i)!)]$$

This combinatorial approach provides an exhaustive identification of control sets. However, it may lead to a huge amount of possible combinations, when n becomes big.

2. Compute Expected Cost due to EDP Exposures

Expected losses due to occurrence of EDP exposures can be estimated using the weighted probability function, the P.E.R.T. method under the Accounting definition, and/or the ranking method.

Under the weighted probability, given an exposure, the probability of its occurrence, and the amount of its damage, the expected loss is defined as follows:

$$l_j = Pr(e_j) \cdot d_j$$

Under the P.E.R.T. method, given an exposure and the smallest($l1_j$), the most likely($l2_j$) and the largest($l3_j$) estimated dollar losses if the exposure occurs, the expected loss is defined as follows:

$$l_j = (l1_j + l2_j + l3_j) / 6$$

The Ranking method is based on two types of subjective rating scales related to the Rank P and the Rank Q. Rank P is the probability of occurrence of computer failures and Rank Q is the amount of damage caused by a potential exposure. Given P and Q, the expected loss can be computed as follows:

$$l_j = 10^{(P+Q-3)} / 4$$

3. Compute the Value of each Control Activity

The value of a control activity a_i is defined as the sum of the products between the expected amount of damage l_j and the effectiveness of a_i on exposure e_j:

$$v_i = \sum_{j=1}^{m} (l_j \cdot f_{ij})$$

4. Compute the Total Value of each Control Set

The calculation of the value of each control set must take into consideration joint effects of multiple control activity on single exposure. For all a_i contained in s_k:

$$V_k = \begin{cases} \sum_{i=1}^{n} \left(\sum_{j=1}^{m} (l_j \cdot f_{ij}) \right) & \text{if } f_{ij} > 0,\ f_{pj} = 0, \\ & \text{for all } p;\ a_p \in s_k \\ + \sum_{i=1}^{n} \left(\sum_{j=1}^{m} (l_j \cdot (1 - \prod_{i=1}^{n} \cdot (1 - f_{ij}))) \right) & \text{if } f_{ij},\ f_{pj} > 0, \\ & \text{for all } i \neq p;\ a_p \in s_k \end{cases}$$

5. Compute the Total Expected Loss for each Control Set

The enforcement of control measures is likely to reduce the probability of occurrence of computer failure and, consequently, the expected loss. However the reduction of expected loss is effective only on the exposures that are affected by controls. The computation of expected losses includes joint effects of control activities. Thus, for all a_i in s_k:

$$L_k = \begin{cases} \sum_{i=1}^{n} \left(\sum_{j=1}^{m} (l_j \cdot f_{ij}) \right) & \text{if } f_{ij} > 0,\ f_{pj} = 0, \\ & \text{for all } p;\ a_p \in s_k \\ + \sum_{i=1}^{n} \left(\sum_{j=1}^{m} (l_j \cdot (1 - \prod_{i=1}^{n} \cdot (1 - f_{ij}))) \right) & \text{if } f_{ij},\ f_{pj} > 0, \\ & \text{for all } i \neq p;\ a_p \in s_k \\ \sum_{j=1}^{m} l_j & \text{else} \end{cases}$$

6. Compute the Cost for each Control Set

The cost of the control set C_k, is the sum of the costs of the individual control activities in the set:

$$C_k = \sum_{i=1}^{n} c_i \qquad \text{if } a_i \in s_k,\ k = 1,S$$

7. Compute the Benefit Cost Ratio for each Control Set

The Cost Benefit Ratio of a control set s_k can be defined as the gross value of s_k(step 4) divided by the total cost of the set(step 6):

$$BCR = V_k / C_k \qquad \text{where } k = 1,S$$

8. Compute Total Expected Cost for each Control Set

The total expected cost for the control set is the sum of the total cost of control C_k plus the total expected loss:

$$TC_k = C_k + L_k \qquad \text{where } k = 1,S$$

9. Select the Optimal Control Set

The determination of an optimal control set depends on the selection criterion adopted by EDP managers or auditors. One can either choose the control set that minimizes the total expected cost(TC_K) or the one that maximizes the Benefit Cost Ratio(BCR). BCR represents the amount of benefits obtained per unit of cost of the investment.

Figure 2.1 represents the whole process of the CEA Model.

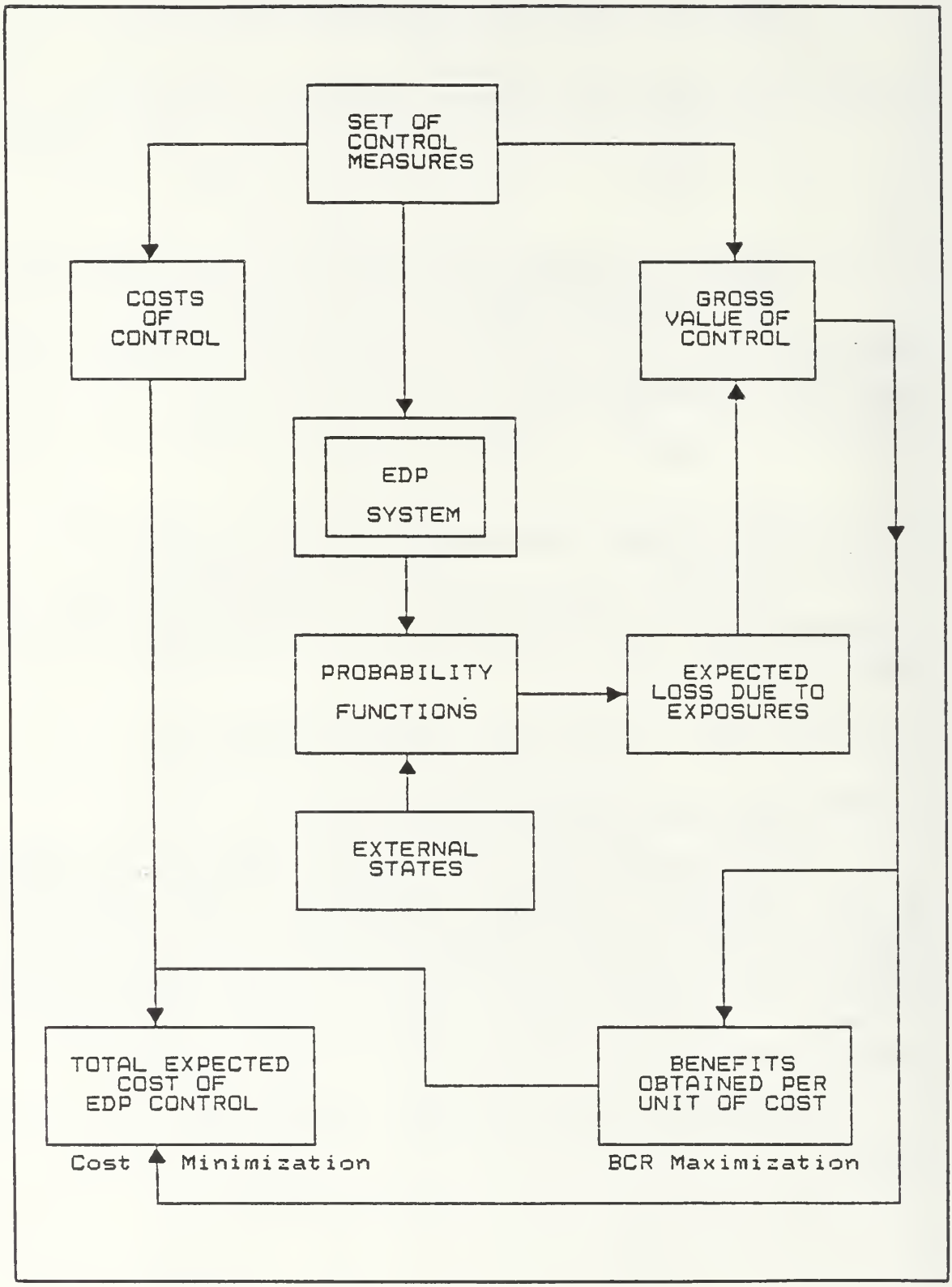


Figure 2.1 The Process of the CEA Model

III. THE DSS FRAMEWORK

The literature on DSS agree upon the emergence of the three main components of a DSS: the Dialog component, the Model component and the Data component. The separation of these components can result in simplicity of development and maintenance [Ref. 5]. Although these advantages are extremely desirable, there are cases where the complexity of the model component makes the complete separation ineffective.

The CEA-DSS falls in that category because the nature of the CEA Model requires a fairly complex and restrictive User Interface. The Quick-hit development strategy, according to which the DSS has been developed, consists of using the latest technology to quickly design a low-cost system for immediate pay-off [Ref. 6].

A. THE ROLES AND FUNCTIONS OF THE CEA-DSS

From the decision maker point of view, the user may expect CEA-DSS to perform the following functions:

- Save substantial amount of time to generate the numerous alternative control combinations.
- Support him or her to evaluate the alternatives and choose among them the alternative that fits better at the particular situation according to the available budget.
- Provide the capability to monitor EDP control and security systems in terms of Cost-Effectiveness.
- Provide graphical and tabular analyses to help the decision maker select close alternatives.

From a system analysis viewpoint , CEA-DSS essentially performs the roles of data analysis and generation of expected costs and benefits of control strategies. Data analysis also allows the decision maker to sort the data.

B. THE SYSTEM RESOURCES

Decision processes are dependent on variations in decision makers, i.e. users, as well as types of problems or tasks. Observations on decision makers indicate that:

- Many users have trouble describing a decision-making process. They seem, instead, to rely on conceptualizations, such as graphs or tables, when making or explaining a decision [Ref. 7]. Thus the DSS must help the user to conceptualize a problem.
- Users need memory aids [Ref. 8]. These memory aids may be physical, such as scratch paper, memos, or reports. The DSS should provide memory aids compatible with their needs. Directories, databases, workspaces, triggers are some typical memory aids the DSS should provide the user.
- Users have different styles, skills and knowledge [Ref. 9]. Therefore, if the DSS is designed to support a specific process, it would probably support a specific set of styles, skills and knowledge.
- Users expect to exercise control over the DSS. Direct control of the DSS allows the DSS to satisfy the different styles mentioned above. The user must understand what the DSS can do and be able to interpret its outputs.

IV. THE DIALOG COMPONENT

The dialog component is the most elegant part of the DSS design. There are no absolute rules or algorithms for the design process. It is often left upon the intuition of the designer to balance user requirements with system requirements and provide the optimal dialog component.

The dialog component of the CEA-DSS consists, at least conceptually, of the following three main units:

- The user interface.
- The intermodule linkage.
- The control.

A. THE USER INTERFACE

The user interface unit provides the link between the user and the system. Its primary concern is to make the system 'user friendly'. Even if a DSS provides extremely powerful functions, it may not be used if the user interface is unacceptable.

For the CEA-DSS a full screen frame is the standard presentation of the system to the end-user. The user, having only one screen format to deal with, gets familiar with the system faster.

The man-machine interaction is carried out through menus, questions/answers, messages, input/output forms, graphics, printed reports and a help facility.

1. The Frame

Figure 4.1 shows the frame of the CEA-DSS. It is divided into the following areas:

- The PROBLEM area. In this area appears the description of the problem currently processed.

- The ACTION area. This area informs the user about which part of the system is currently accessed.
- The WORK area. This is the place where the greatest part of the dialog is accomplished. All the menus, messages, input/output forms and the directory of the DSS appear here.
- The SUBMENU/SELECTION area. In this area appear submenus in line format and the user is asked to make a selection. This area is also reserved for question/answers and the 'press any key..' prompt, reminding the user that the system is waiting for some action.

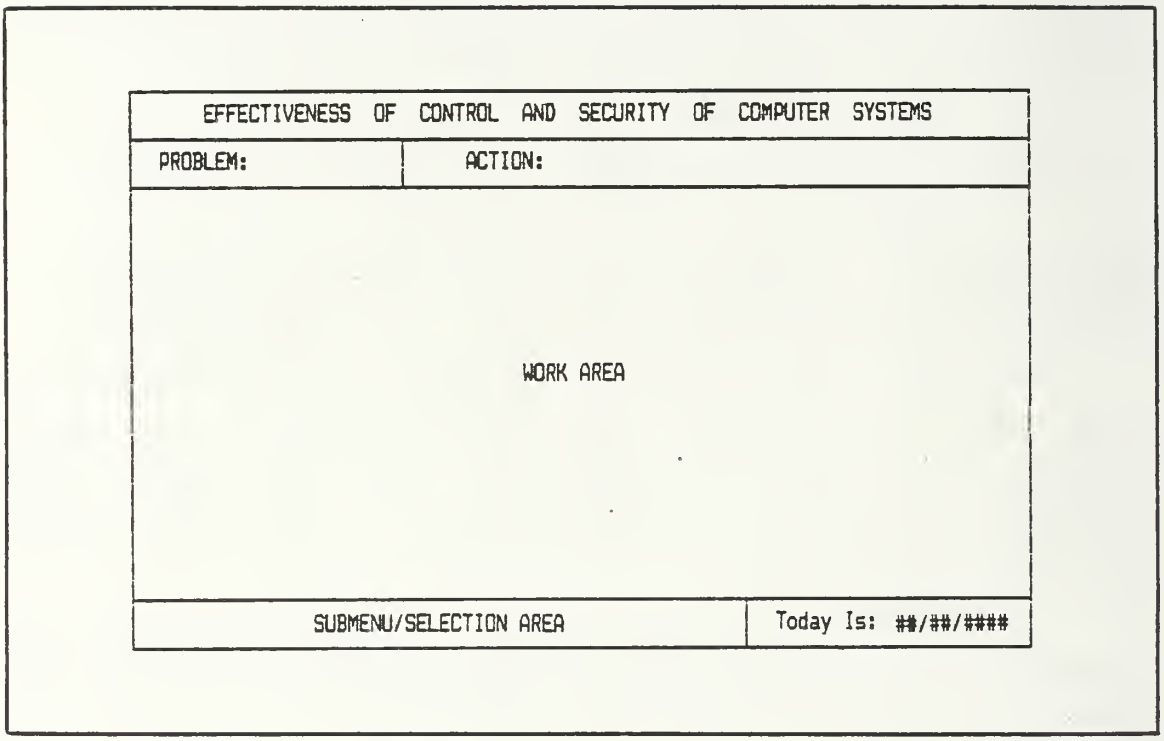


Figure 4.1 The Frame of the CEA-DSS

2. The Menus

The menus of the CEA-DSS are organized in a four level tree hierarchy. The root of the tree is the MAIN MENU of the system. From this menu can be called any menu that belongs in the second level. The latter contains has the DATABASE MENU, the MODEL MENU, and the SENSITIVITY ANALYSIS MENU. The third level consists of the database submenu, the

CONTROL STRATEGY MENU, the GRAPHICS MENU and the PRINT MENU. Finally, in the fourth level there are the control strategy, the graphics and print submenus. Figure 4.2 shows the tree hierarchy of the menus.

One level at a time, upwards or downwards, is allowed for the same branch of the tree. Changes from one branch to another require the control to be routed up to the root of these two branches. Although this is a little restrictive for the user, it improves the indermodular independence and, consequently, the overall control and clarity in the system.

All the menus, submenus not included, have their own help command which the user may use to get some useful information about the area of the DSS he/she is currently accessing. Most of the menus are discussed in Chapter 9.

3. Questions/Answers

There are a few questions/answers in the CEA-DSS. They are used either in cases where the system must be reassured that the user made the correct selection, or for single data entries.

4. Messages

Messages, almost always, appear at the center of the work area accompanied by a 'beep' sound. Messages, according the reason of their initiation, fall into the following three categories:

- Trigger Messages. These remind the user that certain operations may need to be performed that the system cannot accomplish.
- Informal messages. They inform the user about what process is the system performing. The primary concern of this category is to cover the gaps in the dialog caused by time consuming processes.
- Error Messages. They are initiated when the user supplies the system with incorrect entries. While editing exposures or controls, 'beep' sounds notify the user for entry errors.

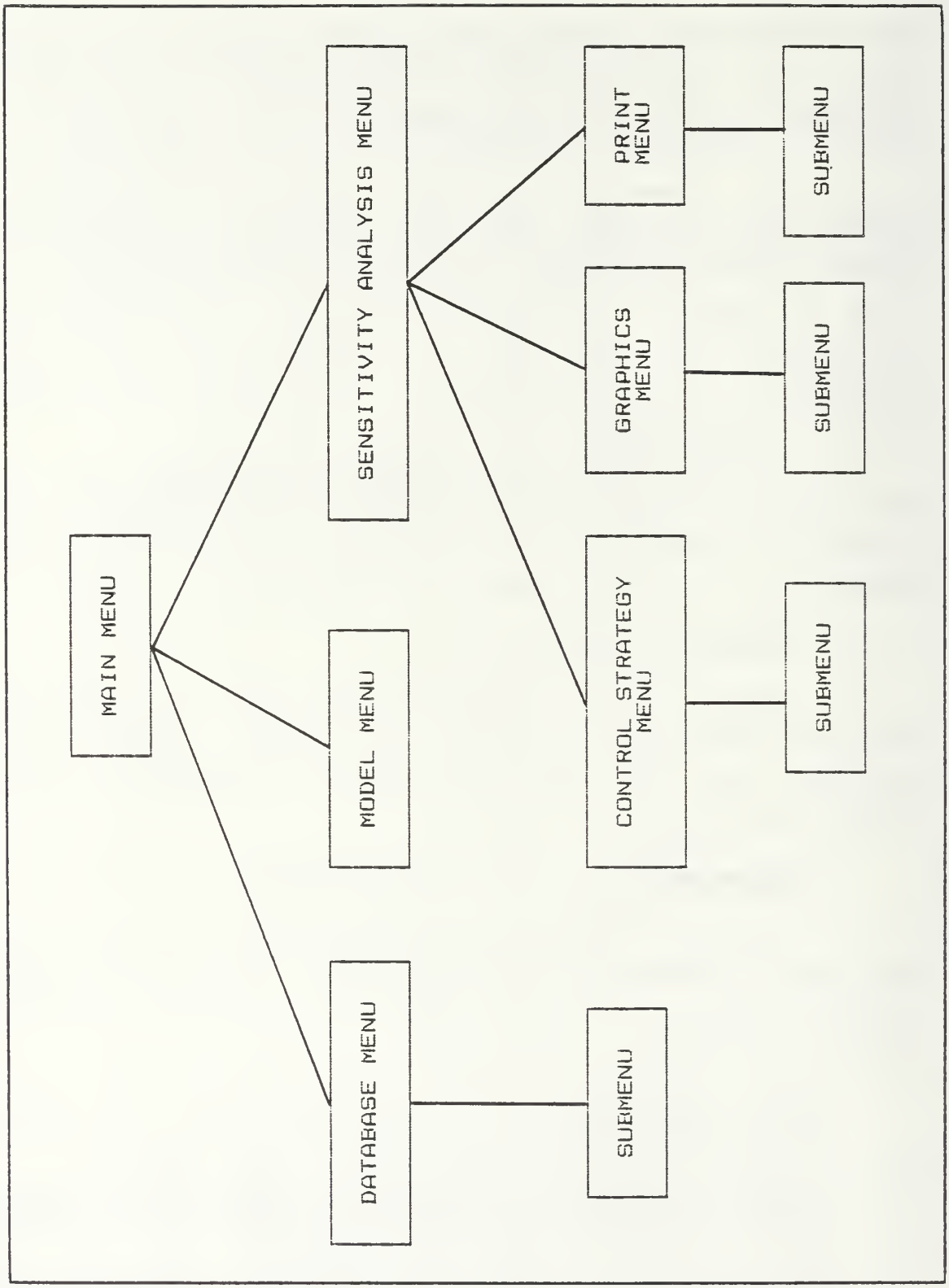


Figure 4.2 Menus' Tree Hierarchy

All messages along with the reason that causes their initiation are listed in Appendix A.

5. Input/output Forms

The system needs only three forms for its operation. Two of them are input/output forms and one output form. One input/output form is dedicated to the Control activities and the other one to the Exposures. Both are used by the Database Management System for editing purposes. The output form is used by the Sensitivity Analysis for presenting the most effective or most cost effective Control Strategy.

Figure 4.3 shows the two input/output forms. Fields filled with Xs indicate that any character is valid, while 9s represent numeric characters only. Notice that the control input/output form is a variable one. The number of the "Effectiveness on Exposure" fields that appear on the form depends on the number of Exposures.

6. Graphics

The objective of the graphics part is to help the user conceptualize the differences among alternatives over the cost range he/she prefers. Graphics can also be used to supply parameters for the operations. For example, a point selected on a graph can identify a key value that will be used to retrieve detailed information. Representations like curves and histograms are the most appropriate for this particular application.

7. Printed Reports

Although not technically a part of the DSS, printed reports are aimed to provide the user with an easy-to-read summary of the processed problem. This summary consists of the exposure table, the control table and the listing of the sets generated by the model. The user may select any of these reports or all of them to be printed.

```
PROBLEM: PROBLEM1          ACTION: ADD EXPOSURE

Index:01     Description:XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

 WEIGHTED: Damage:$99999999     Probability:0.999

 P.E.R.T:  Smallest:$99999999   Most Likely:$99999999   Largest:$99999999

 RANKS:    Rank P:9.999         Rank Q:9.999
 Rank P Damage caused by error          Rank Q Damage caused by failure
   0   virtually impossible               0   negligible
   1   might happen once in 400 years     1   about         $10
   2   might happen once in  40 years     2   about        $100
   3   might happen once in   4 years     3   about      $1,000
   4   might happen once in 100  days     4   about     $10,000
   5   might happen once in  10  days     5   about    $100,000
   6   might happen once in   1   day     6   about  $1,000,000
   7   might happen ten  times a  day     7   over   $1,000,000

IS RECORD CORRECT(Y/N)? :                 Today Is: 8/19/1985
```

```
PROBLEM: PROBLEM1          ACTION: ADD CONTROL

Index:01     Description:XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
                  Cost:$99999999

Effectiveness on Exposure  1: 0.999     Effectiveness on Exposure 13: 0.999
Effectiveness on Exposure  2: 0.999     Effectiveness on Exposure 14: 0.999
Effectiveness on Exposure  3: 0.999     Effectiveness on Exposure 15: 0.999
Effectiveness on Exposure  4: 0.999     Effectiveness on Exposure 16: 0.999
Effectiveness on Exposure  5: 0.999
Effectiveness on Exposure  6: 0.999
Effectiveness on Exposure  7: 0.999
Effectiveness on Exposure  8: 0.999
Effectiveness on Exposure  9: 0.999
Effectiveness on Exposure 10: 0.999
Effectiveness on Exposure 11: 0.999
Effectiveness on Exposure 12: 0.999

IS RECORD CORRECT(Y/N)? :                 Today Is: 8/19/1985
```

Figure 4.3 Input/output Forms

8. Help

The purpose of the help facility is to provide the user with on-line information about the specific area of the system he/she is currently accessing. Each help, one for each menu, is written in such a level of detail that enables its presentation in one full screen frame only. All help documents appear in Appendix B.

B. THE INTERMODULE LINKAGE

This unit assures the liaisons with the model and the data component. Usually, it is maintained by a set GOTO, CASE and IF_THEN_ELSE statements. Its nature and structure are highly dependent on the programming language and the hardware configuration being used for the CEA-DSS.

C. THE CONTROL

On the one hand, as in section 3.B stated, users expect to exercise control over the DSS. On the other hand, the system has to control its processes to assure an error free operation, not affected by incorrect entries and requests. The control unit is the part of the dialog component which bridges these two requirements. It is the filter between the user interface and the intermodule linkage unit. Validation of input data and verification of user requests are its primary functions. All the error messages are initiated by this unit. Finally, it can be stated that the control unit provides the boundaries within which the user is allowed to control the process.

V. THE MODEL COMPONENT

The most important units of the model component are the Model Base,the Model Base Management, the Model execution, the Sensitivity Analysis, the Dialog Interface and the Data Interface.

A. THE MODEL BASE

The following five routines, required for the CEA Model, are the content of the model base for the CEA-DSS. (The mathematical definition of these methods was discussed in section 2.C).

1. The Weighted Method

This routine computes the expected cost due to EDP exposures using the weighted probability function. It retrieves the required data, directly from the data base, manipulates the data and stores the results in memory for subsequent computations.

2. The P.E.R.T. Method

It is exactly the same with the Weighted Method routine except that it uses the P.E.R.T. method to compute the expected cost due to EDP exposures.

3. The Ranking Method

Similar to the others, it computes the expected cost due to EDP exposures using the Ranking Method.

4. The Effective Control

The role of this routine is twofold: To compute the Value of each Control activity and, if possible, to reduce

the number of the control activities that will be actually used in the generation of the control sets. The routine, having the results of one of the tree methods, retrieves, directly from the database, data related to the Control activities. For each Control, it computes first the value and then, it compares that value with the associated cost. If the value is greater that the cost, the result is sent to a secondary storage for subsequent computations. If the value is less or equal to the cost, the Control activity is ignored.

5. The Control Sets

The output of the Effective Control routine is used by the Control Sets to generate the control sets. For each control set it computes the steps 4 to 8 described in the CEA model. If the Total Value of the set is greater than its cost, the set is stored in the database for decision analyses support, otherwise it is ignored.

B. THE MODEL BASE MANAGEMENT

The role of the Model Base Management is to coordinate the model base and the data analysis functions. Since the CEA-DSS is aimed to support only the model described in Chapter II, the Model Base Management does not provide for on-line modeling or model update and restructure.

Its most important function is to enable the user to utilize the model base fully for decision support and to perform analysis of the results. This function is performed by iterative rerun of the model.

Also, it is responsible to update the Problem record, kept in the directory of CEA-DSS, with key information about the model runs. Thus, any future reference to this problem will not require any model execution, except if modifications take place on the initial data or on the cost range.

C. MODEL EXECUTION

Contains statements to call routines from the model base. It controls the execution of the model assuring the logical sequences of computation.

D. SENSITIVITY ANALYSIS

The Sensitivity Analysis unit helps the user analyze the results of the model runs. It is directly controlled by the dialog component. This unit consists of all the routines associated with graphic representations, control strategy selection and hard copy reports.

Input data for the sensitivity analysis are the control sets in the set files. As stated earlier, a model run may produce thousands of control sets. Therefore, it is usual several control sets to have exactly the same cost. Since the amount of data is huge and the analysis is primarily based on costs, the control sets in a set file must be indexed on their cost. This creates the requirement for the database system to provide for direct file access and to allow the existence of duplicate keys within the same index.

E. DIALOG INTERFACE

The model component is directly interfaced with the dialog component in order the user to gain control over its processes. He/she is able to select the desired statistical method and cost range for a model run and the cost range for the data analysis process.

F. DATABASE INTERFACE

The model component is directly interfaced with the data component. This enables the model component to create and delete the set files where the generated control sets are stored.

VI. THE DATA COMPONENT

The data component consists of two main units. The the Database Management System and the Database discussed in the next chapter.

A. THE DATABASE MANAGEMENT SYSTEM (DBMS)

The complexity of the Dialog component and the Model component, as well as the effective and efficient operation of the system lead to the selection of a Relational Database system. One characteristic of a Relational Database is the use of fixed length records. However, variable length records cannot be avoided. Since the data component requires functions like addition, deletion and modification on data, the elimination of modification anomalies seems to be of high priority.

The DBMS provides capabilities for sequential, indexed sequential and direct file access. Indexes are organized as B-trees. In a B-tree, a data unit is accessed by using a key. Any given key, primary key, is related to one and only one data unit in a data file. The system permits also the existence of duplicate keys or secondary keys, which are of great importance for the sensitivity analysis as discussed in the previous section.

B. FILES USED BY THE SYSTEM

Files in the system can be divided into three categories, according to their initial creation:

- Files created by the data component. These are the directory of the system and its index. The directory contains all the problems available in the system's library indexed on their description. Duplicate problem description is not permitted.

- Files initiated by the user. The Controls file and the Exposures file fall in that category, indexed on their 'index'. Index is a unique key generated by the DBMS for management purposes. It keeps track of modification anomalies and makes the user's work easier. Actually, it identifies the current position of the data unit in the data file and NOT the data unit itself.

- Files created by the model execution. Each time the model is executed for a specific method, a set file is created indexed on set cost. Duplicate keys are necessary here because it is possible several sets to have the same cost. These files cannot be modified by the user or the system.

C. FILE CREATION/RETRIEVAL

The Data component has the flexibility to deal with library of problems and not with only one problem. In order to achieve that, it must have the ability to recognize and retrieve the files related to the problem in request, or to create files for that problem, if it is not found in the directory of the CEA-DSS. The algorithm followed is the following:

- The directory of the system has the fixed file name 'PROBLEM'. The data file has the fixed filetype 'DTA'and its index the 'IDX'.

- All the files created for one problem have as file name the description of the problem.

- The controls file has as filetype the 'DCL' and its index the 'ICL'.

- The exposures file has as filetype the 'DXP' and its index the 'IXP'.

- For the set files the algorithm used is more complicated. Additionally, the DBMS must be provided with an identifier indicating the method to which the set file refers. For that reason, the filetype for set files is separated into to fields. The first one, one character long, identifies the method, and the second one, two characters long, identifies the data file or the index. For the first field, the letters 'W','P' and 'R' correspond to the Weighted,Pert and Ranking method. For the second field, the 'DT' denotes the data file and the 'IC' the index file.

VII. DATABASE DESIGN

To some extent, Database design is an intuitive and artistic process. There is no algorithm for it. Typically, it is an iterative process. During each iteration, the goal is to get closer to an acceptable design. The database design is divided into two phases: logical design, where the needs of user are specified, and the physical design, where the logical design is mapped into the constrains of particular program and hardware products.

A. LOGICAL DATABASE DESIGN

1. Logical Database Records

The database of the CEA-DSS is required to maintain four different kinds of records. The first one, the PROBLEM record, is the data unit of the system's directory. Each problem has its own unique record. This record, except the problem description, contains key information about the most recent execution of the model on that problem. The second, is the EXPOSURE record. This record contains the description of the exposure and weights for the three methods. The third, the CONTROL record, has the description, the associated cost and elements indicating the effectiveness of the control activity on different exposures. The last, the SET record, is the output of the model execution and contains the combination of the control activities, and the results of the model run. Field descriptions for the logical database records are shown in Table 2.

Constraints on data items appear on Table 3. These constraints are limitations on the values that database can have. They are divided into three groups. Field constraints limit the values that a given data element can have.

TABLE 2

LOGICAL DATABASE RECORDS

Field	Description
PROBLEM Record:	
Problem_Description	Alphanumeric, 8 characters
Problem_Creator	Alphabetic, 25 characters
Problem_Date	Format MM/DD/YY
Controls_for_ Weighted_Method	Numeric(integer), 2 digits
Controls_for_ P.E.R.T._Method	Numeric(integer), 2 digits
Controls_for_ Ranking_Method	Numeric(integer), 2 digits
Weighted_Method_ Total_Cost_of_Controls	Numeric(integer), 10 digits
P.E.R.T._Method_ Total_Cost_of_Controls	Numeric(integer), 10 digits
Ranking_Method_ Total_Cost_of_Controls	Numeric(integer), 10 digits
EXPOSURE Record:	
Exposure_Description	Alphanumeric, 50 characters
Exposure_Damage	Numeric(integer), 8 digits
Exposure_Probability	Numeric(real), 5 digits
Smallest_Damage	Numeric(integer), 8 digits
Most_Likely_Damage	Numeric(integer), 8 digits
Largest_Damage	Numeric(integer), 8 digits
Exposure_RankP	Numeric(real), 5 digits
Exposure_RankQ	Numeric(real), 5 digits
CONTROL Record:	
Control_Description	Alphanumeric, 50 characters
Control_Cost	Numeric(integer), 8 digits
Control_Effectiveness_ on_Exposure	Numeric(real), 5 digits
SET Record:	
Set_combination	Numeric(binary), variable
Expected_Benefits	Numeric(integer), 10 digits
Expected_Loss	Numeric(integer), 10 digits
Set_Cost	Numeric(integer), 10 digits
Expected_Cost	Numeric(integer), 10 digits
Benefit_Cost_Ratio	Numeric(real), 5 digits

TABLE 3

CONSTRAINTS FOR LOGICAL DATABASE RECORDS

Field_Constraints:

Problem_Description must not be null

Controls_for_Weighted_Method must not be 0

Controls_for_P.E.R.T._Method must not be 0

Controls_for_Ranking_Method must not be 0

Exposure_Probability must be from 0.000 to 0.999

Exposure_RankP must be from 0.000 to 7.000

Exposure_RankQ must be from 0.000 to 7.000

Control_Effectiveness_on_Exposure from 0.000 to 0.999

Benefit_Cost_Ratio must be greater than 1.000

Intrarecord Constraints:

Most_Likely_Damade greater than Smallest_Damage

Largest_Damage greater than Most_Likely_Damage

Interrecord Constraints:

Problem_Description must be unique

Exposure_Description may be unique

Control_Description may be unique

The number of Controls_for_Weighted_Method fields must be equal or less than the number of Control records. The same must be true for the Controls_for_P.E.R.T and Ranking_Method.

The number of Control_Effectiveness_on_Exposure fields must be equal to the number of Exposure records.

The level of the Set_Combination must be equal or less than the number of Control records.

Intrarecord constraints limit values between fields within a given record. Interrecord constraints limit values between fields in different records [Ref 10].

2. Logical Database Record Relationship

Figure 7.1 shows possible relationships among the records used by CEA-DSS. This figure is a data structure diagram. Single/double arrow notation is used to express a one-to-many relationship and double/double arrow represents a many-to-many relationship.

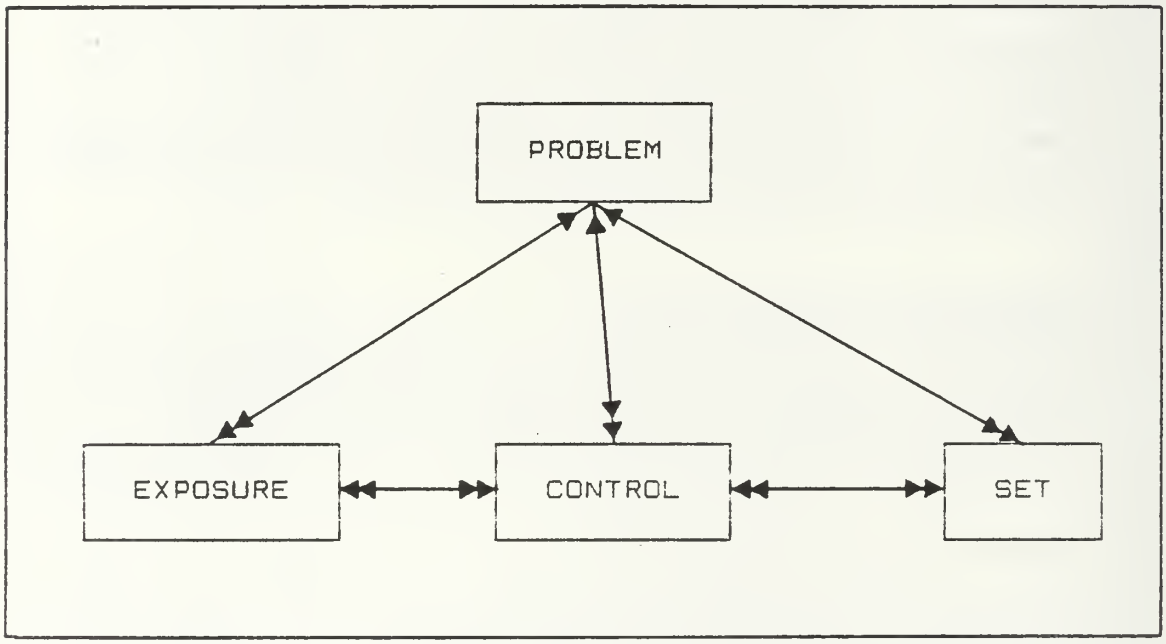


Figure 7.1 Data Structure Logical Diagram

The above complex network is further decomposed into trees in order the database to be able to deal with the data requirements. Figure 7.2 shows the decomposition of the complex network. It is a four level tree structure and represents relationships according to the model specifications. For clarity purpose, the Exposure is represented with the letter 'E' and the Control with the letter 'C'.

The dashed lines connecting sets with controls and controls with exposures indicate that it is not necessary for a set to include all the control activities or a control activity to influence all the exposures.

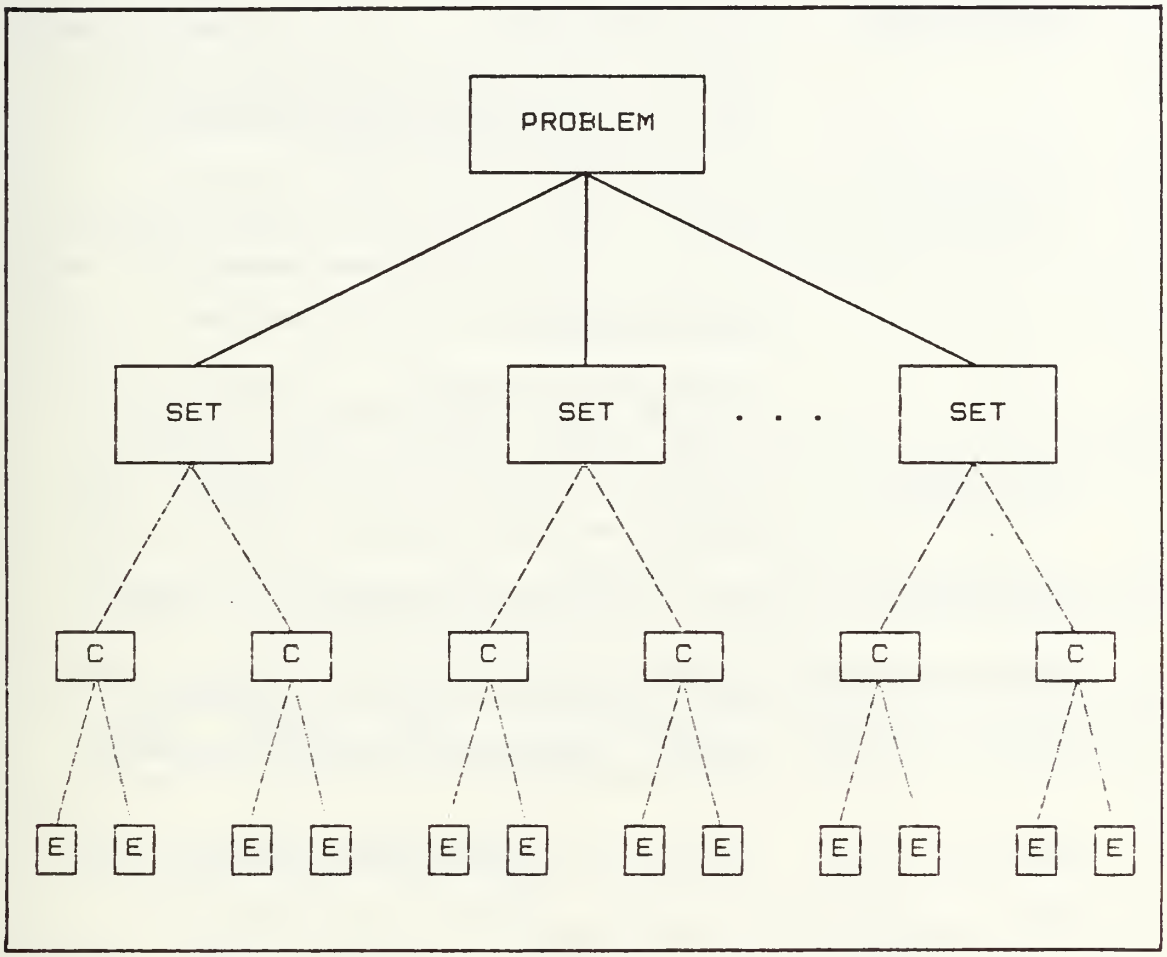


Figure 7.2 Decomposition of the Data Structure

3. Data Manipulation in the CEA-DSS Database

The possible transactions and the data that the transactions can change are listed in Table 4. Some transactions change data, some add new data, some delete data and some are simple queries. Queries are all the transactions in the sensitivity analysis part. No data are modified.

TABLE 4

POSSIBLE TRANSACTIONS FOR THE CEA-DSS

Transaction	Data or Transaction Involved
NEW PROBLEM	Add one record in the directory
	Create Control and Exposure files
	Add, at least, two Exposures and two Control activities
DELETE PROBLEM	Erase Control and Exposure files
	Erase any existing Set File
	Remove the Problem record from the directory
	Erase the directory, if there is not another problem in it
ADD EXPOSURE	Add one record in the Exposure file
	Update records in the Control file
ADD CONTROL	Add one record in the Control file
DELETE EXPOSURE	Remove record from the Exposure file, if it has more than two records
	Remove references to this Exposure from the Control records
DELETE CONTROL	Remove record from the Control file, if it has more than two records
EDIT EXPOSURE	Modify record in the Exposure file
EDIT CONTROL	Modify record in the Control file
MODEL EXECUTION	Erase any existing Set file for the selected method.
	Create Set file
	Add Control Sets in the Set file
	Update record of the current problem in the directory of the system

B. PHYSICAL DATABASE DESIGN

During the second phase of the database design, the physical design, a transformation takes place. The logical schema is transformed into the particular data constructs that better satisfy the implementation requirements and constraints.

1. Design Constraints

One implementation requirement for the CEA-DSS is to be used on microcomputers. This requirement along with the other requirements, discussed in the framework, introduce the following constraints for the physical database design phase:

- Integer numbers are not allowed in the system. All numbers have to be of type real and will be stored in the system as strings of characters.
- The length of records in bytes must be limited as much as possible because of microcomputer limitations.
- Since the size of the Control record depends on the number of the Exposure records, the number of Exposures for one problem may be 24 at maximum.
- The number of control activities for one problem are limited to 13 at maximum. Three model runs, one for each method, for a problem having 13 control activities, may generate up to 24,576 set records. These records need at least 3 Mbytes to be stored.

2. The Physical Schema

The Physical database records are slightly differentiated from logical records to satisfy the design constraints. The field description of the records is shown on Table 5 where all numerics are of type real and the abbreviation 'char' instead of 'character' is used.

Keys are identified according to the data retrieval requirements. The record relationships and constraints emain the same as in the logical design.

The idea of having flat files in the database is infeasible because of the model's computational complexity.

TABLE 5

PHYSICAL DATABASE RECORDS

Field	Description
PROBLEM Record: Indexed on Problem_Description	
Problem_Description	Alphanumeric, 8 char
Problem_Creator	Alphabetic, 25 char
Problem_Date	Format MM/DD/YY
Controls_for_ Weighted_Method	Array(1..13) of 2 char
Controls_for_ P.E.R.T._Method	Array(1..13) of 2 char
Controls_for_ Ranking_Method	Array(1..13) of 2 char
Weighted_Method_ Total_Cost_of_Controls	Numeric, 10 char
P.E.R.T._Method_ Total_Cost_of_Controls	Numeric, 10 char
Ranking_Method_ Total_Cost_of_Controls	Numeric, 10 char
EXPOSURE Record: Indexed on Exposure_Index	
Exposure_Index	Numeric, 2 char
Exposure_Description	Alphanumeric, 50 char
Exposure_Damage	Numeric, 8 char
Exposure_Probability	Numeric, 5 char
Smallest_Damage	Numeric, 8 char
Most_Likely_Damage	Numeric, 8 char
Largest_Damage	Numeric, 8 char
Exposure_RankP	Numeric, 5 char
Exposure_RankQ	Numeric, 5 char
CONTROL Record: Indexed on Control_Index	
Control_Index	Numeric, 2 char
Control_Description	Alphanumeric, 50 char
Control_Cost	Numeric, 8 char
Control_Effectiveness_ on_Exposure(1..24)	Numeric, 5 char
SET Record: Indexed on Set_Cost	
Set_combination	Array(1..13) of 2 char
Expected_Benefits	Numeric, 10 char
Expected_Loss	Numeric, 10 char
Set_Cost	Numeric, 10 char
Expected_Value	Numeric, 10 char
Expected_Cost	Numeric, 10 char
Benefit_Cost_Ratio	Numeric, 5 char

More specifically, the use of flat files should increase dramatically the time required for a model run, something undesirable for a DSS.

Variable length records are used instead. This variability in length results in loss of storage capacity because the record occupies space equal to its maximum length regardless its actual length. This, off-course, is the primary disadvantage of the variable length records, but for that particular application is justified by the fact of time savings.

VIII. IMPLEMENTATION OF THE CEA-DSS

One of the objectives of the implementation phase is to use the CEA-DSS with microcomputers.

A. THE PROGRAMMING LANGUAGE

The complexity of dialog and data component underline the need for a structured programming language which can support character manipulations, screen management and, to some degree, mathematic calculations. Turbo Pascal (Version 2.0) was chosen for this particular implementation.

B. SUPPORTING PACKAGES

Turbo Access Toolbox (Version 1.00) is used for the database management system. Turbo Access provides for sequential, indexed sequential and direct file access, allowing and the existence of duplicate keys in an index file. Turbo Graphix Toolbox (Version 1.00A) is used for the graphics part of the system.

C. THE DATA FLOW IN THE CEA-DSS

In order to deal with the high complexity of the data and transaction flow, it was necessary to divide the system from the beginning into four major areas. This helped to draw the initial diagrams. Using these diagrams as the base, after reviews and refinements, the final software structure was derived. These four areas are the following:

1. The Main Area

This area contains data flows and transactions occurring from the initialization of the system until the

main menu appear on the screen and the user make his/her selection. Figure 8.1 shows the refined flow diagram of the main area.

2. The Database

Figures 8.2, 8.3 and 8.4 are the flow diagrams of this area. It contains transactions and data flows related to the database management system, like updating control and exposure files, switching problems, and deleting problems.

3. The Model

The model area diagram, Figure 8.5, describes all the operations of the model execution. Figure 8.6, presents in detail the data flow during the generation of the control sets. This is the most important and most complex part of the CEA model and is included here for maintenance and future modification or improvement purposes.

4. The Sensitivity Analysis Area

Transactions and data flows associated with the decision support part of the CEA-DSS are illustrated in Figures 8.7, 8.8 and 8.9.

D. SOFTWARE STRUCTURE

The refined software structure, Figure 8.10, is a rearrangement of the flow diagrams from the perspective of the flow of control in the system. The requirement for the user to access control over the whole process, underlines the need for a hierarchical flow of control among the various processes of the system. Top-down is considered as the most effective design for the CEA-DSS since it results in a modular and highly cohesive software structure. Modularity and high cohesion facilitate the coding and maintenance phases.

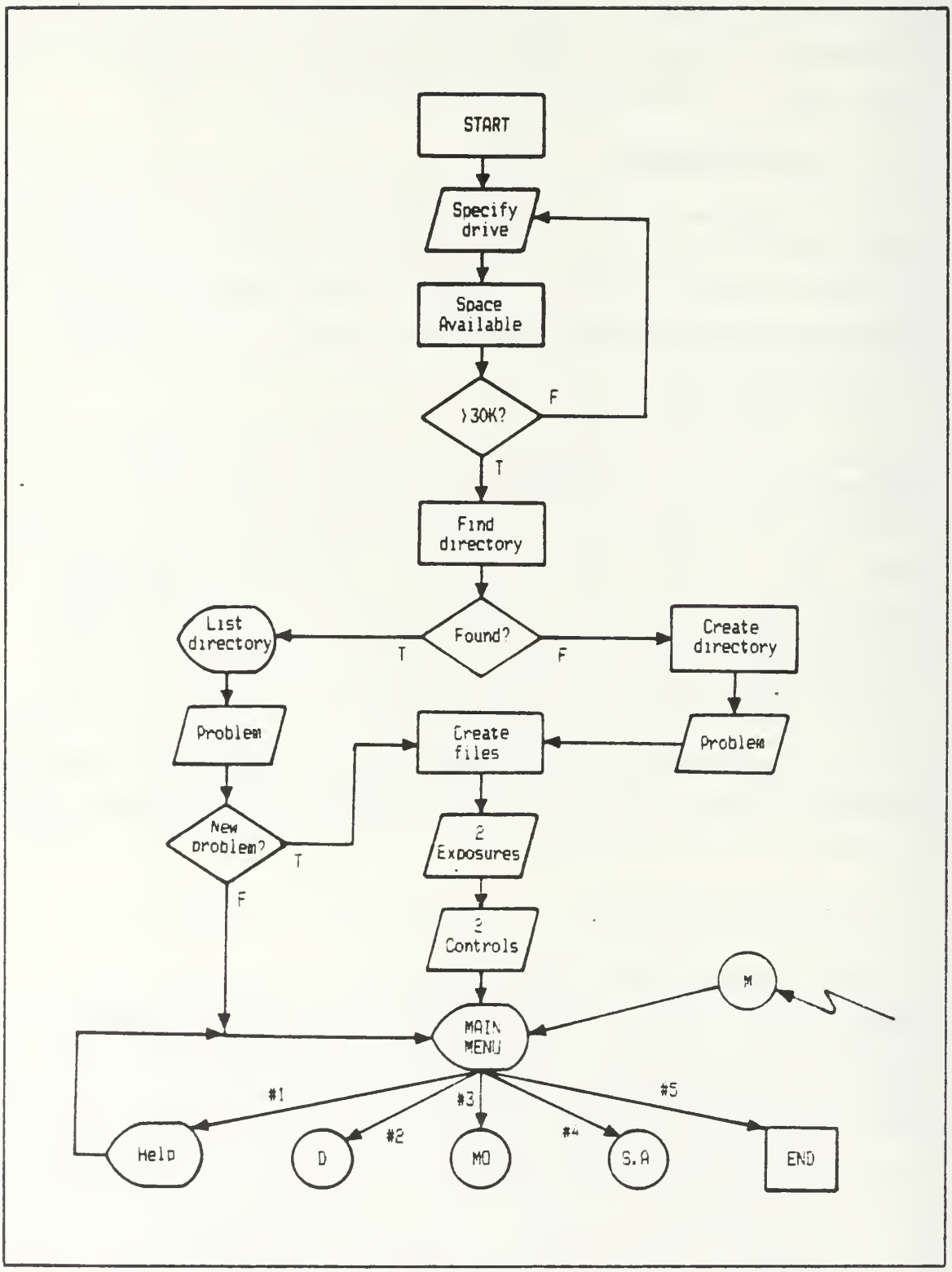


Figure 8.1 Main Area Flow Diagram

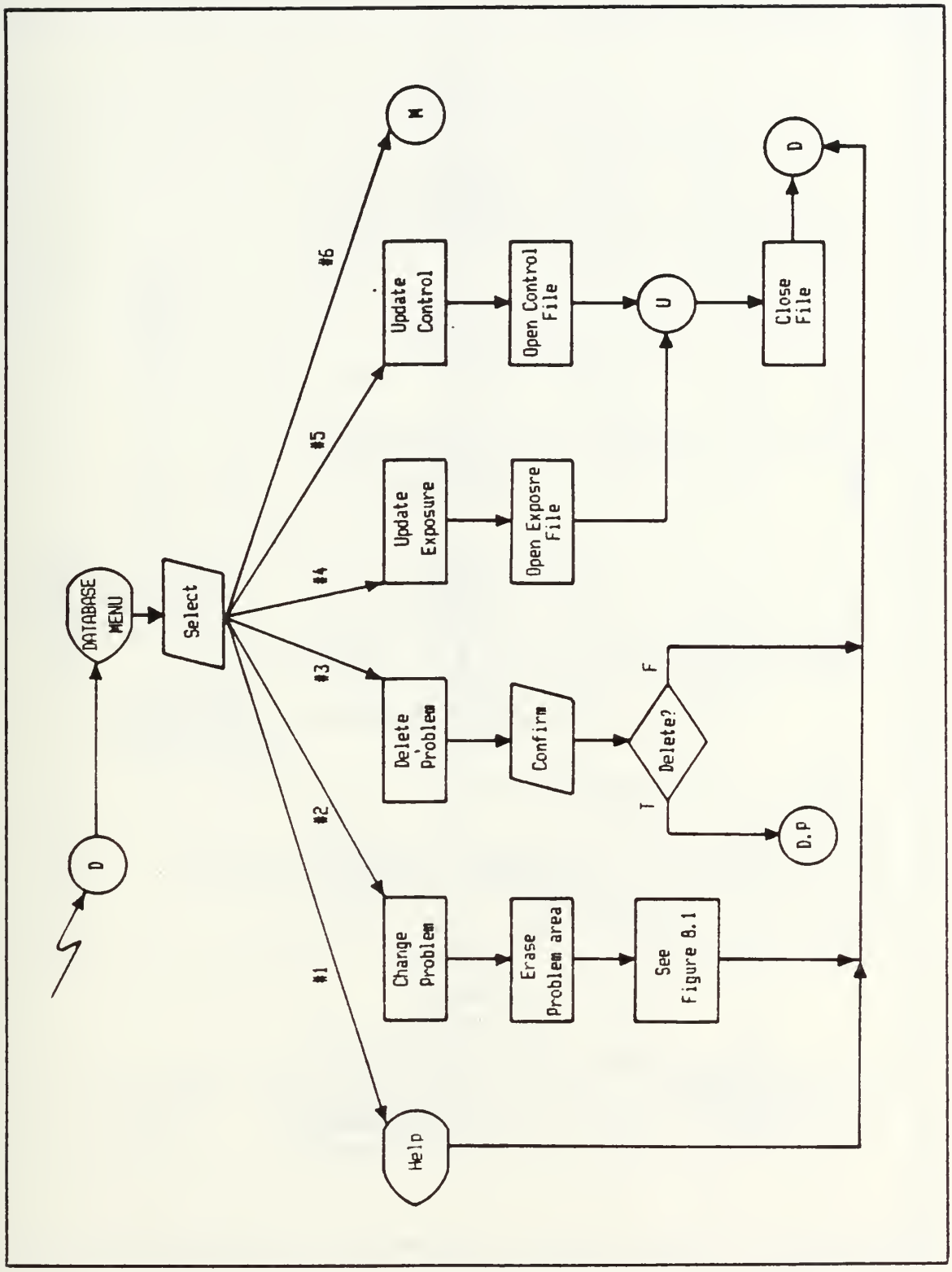


Figure 8.2 Database Flow Diagram

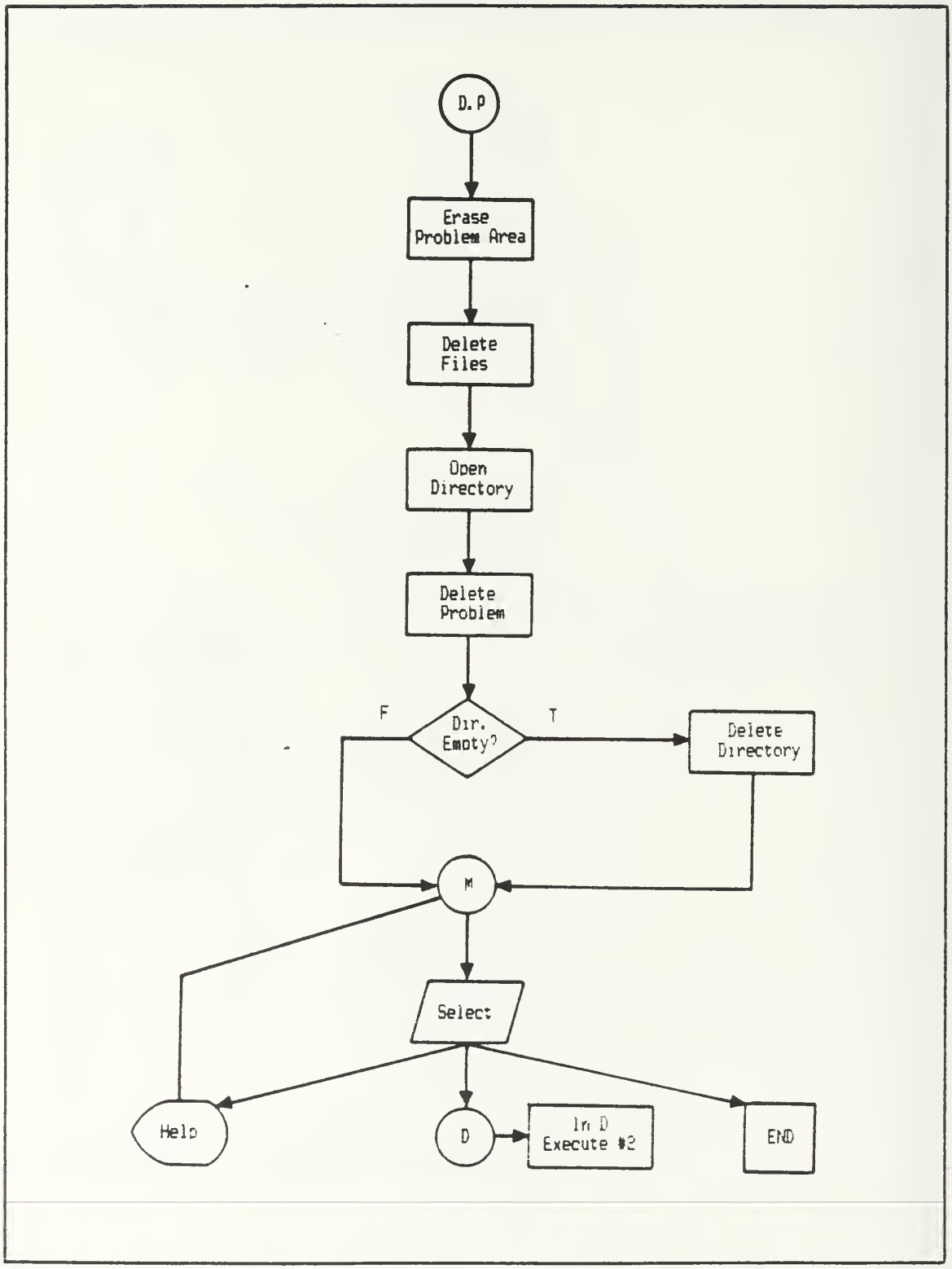


Figure 8.3 Delete Problem Flow Diagram (Database)

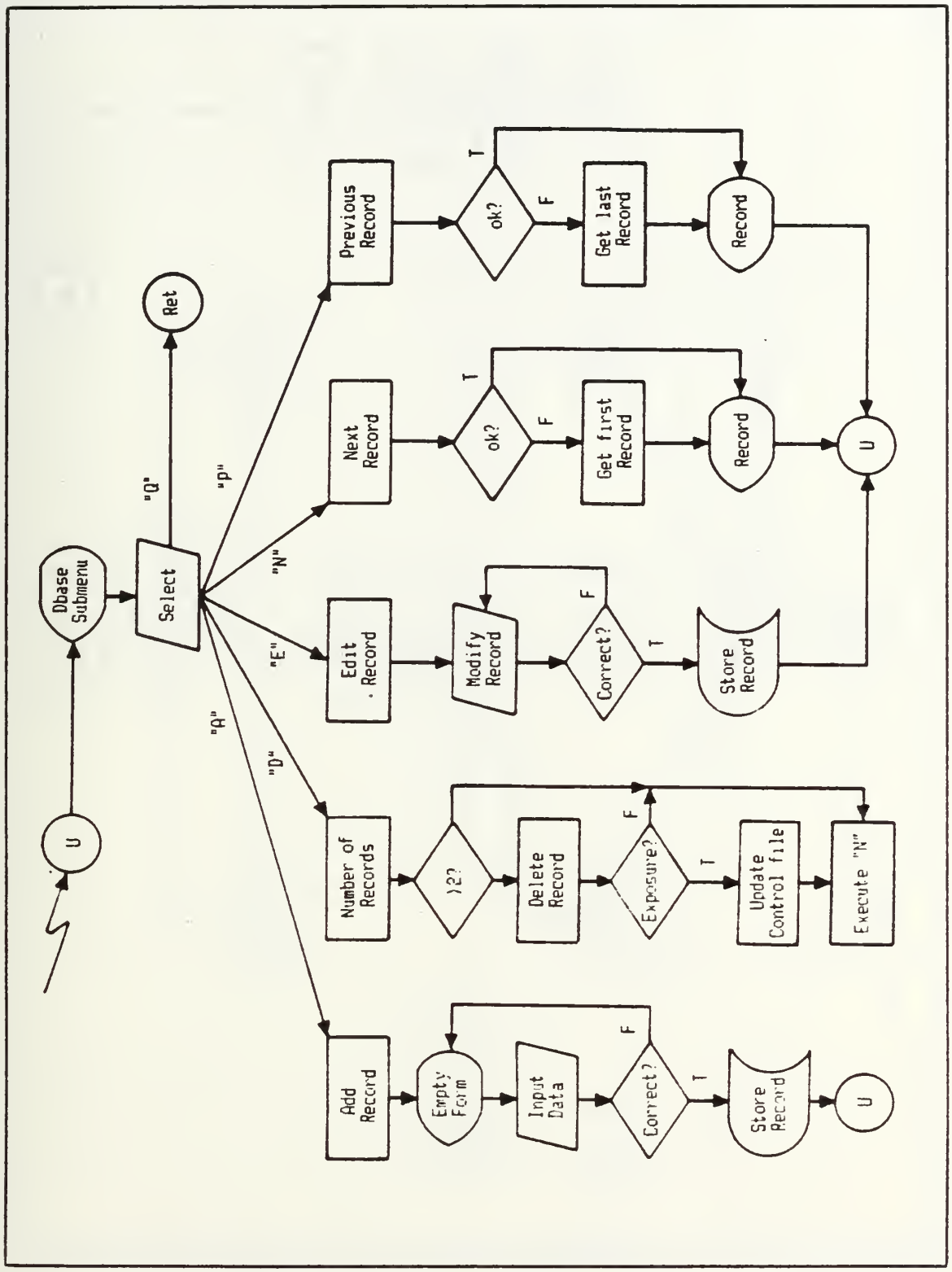


Figure 8.4 Update Files (Database)

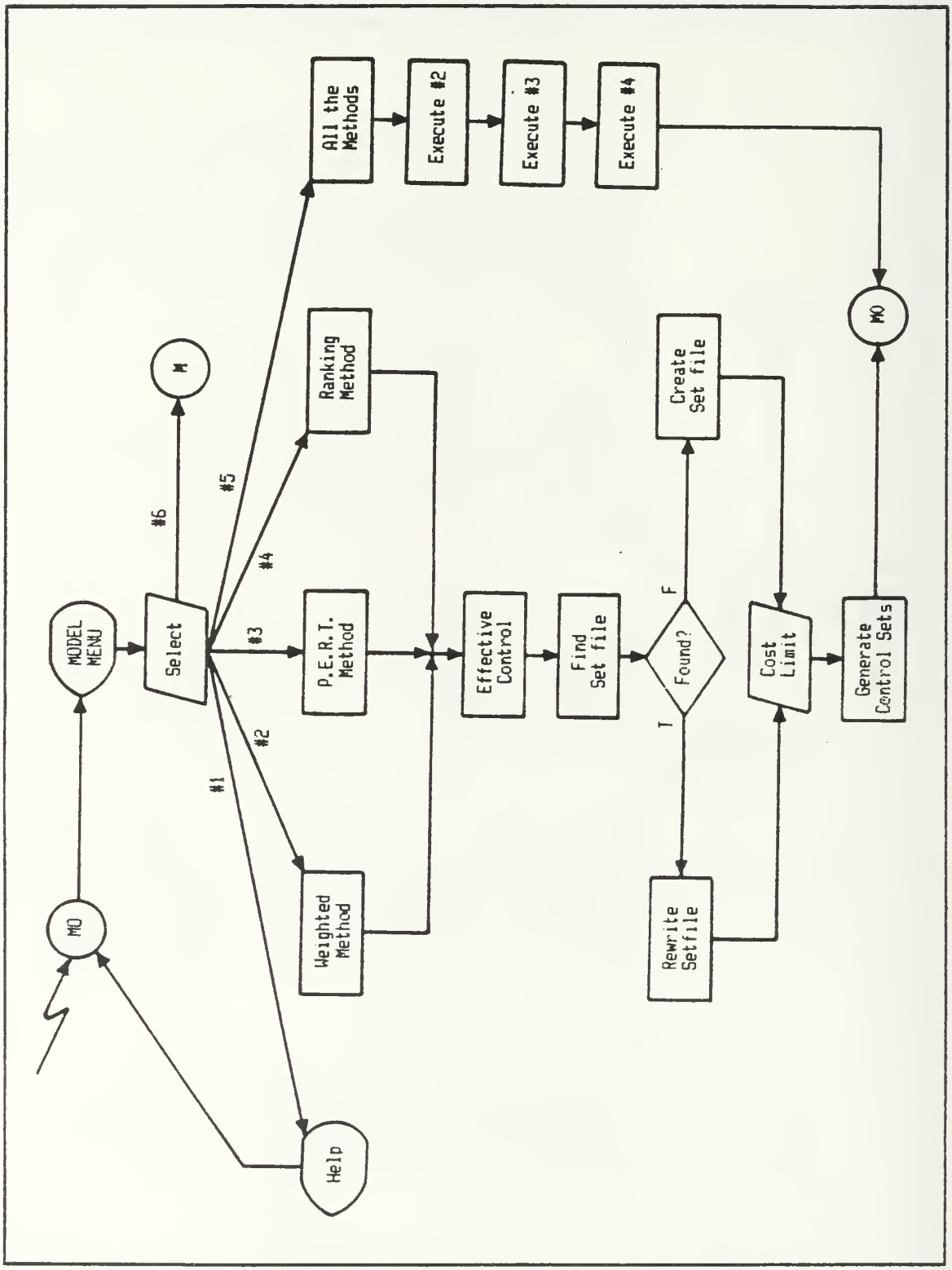


Figure 8.5 Model Flow Diagram

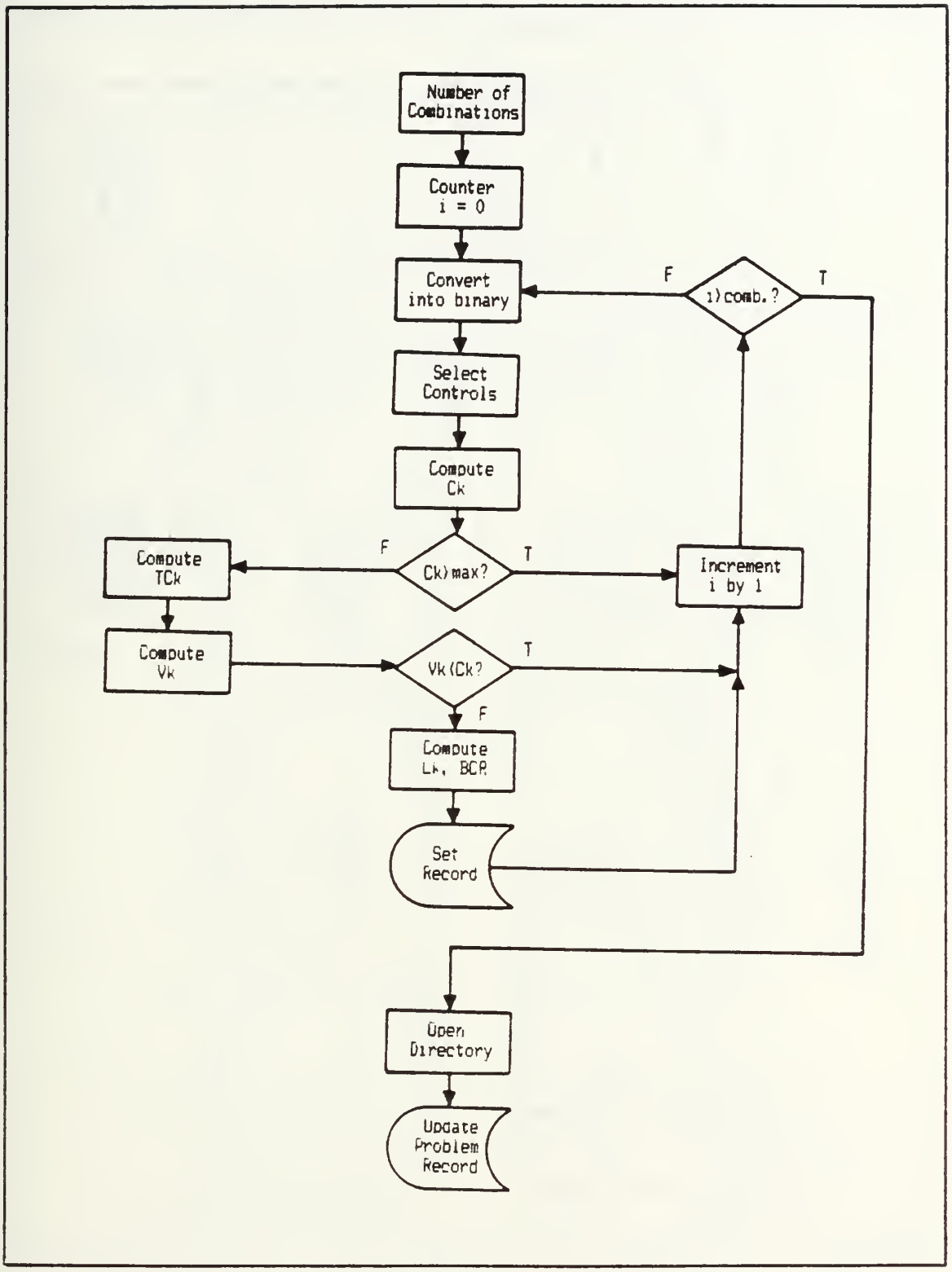


Figure 8.6 Control Sets Flow Diagram (Model)

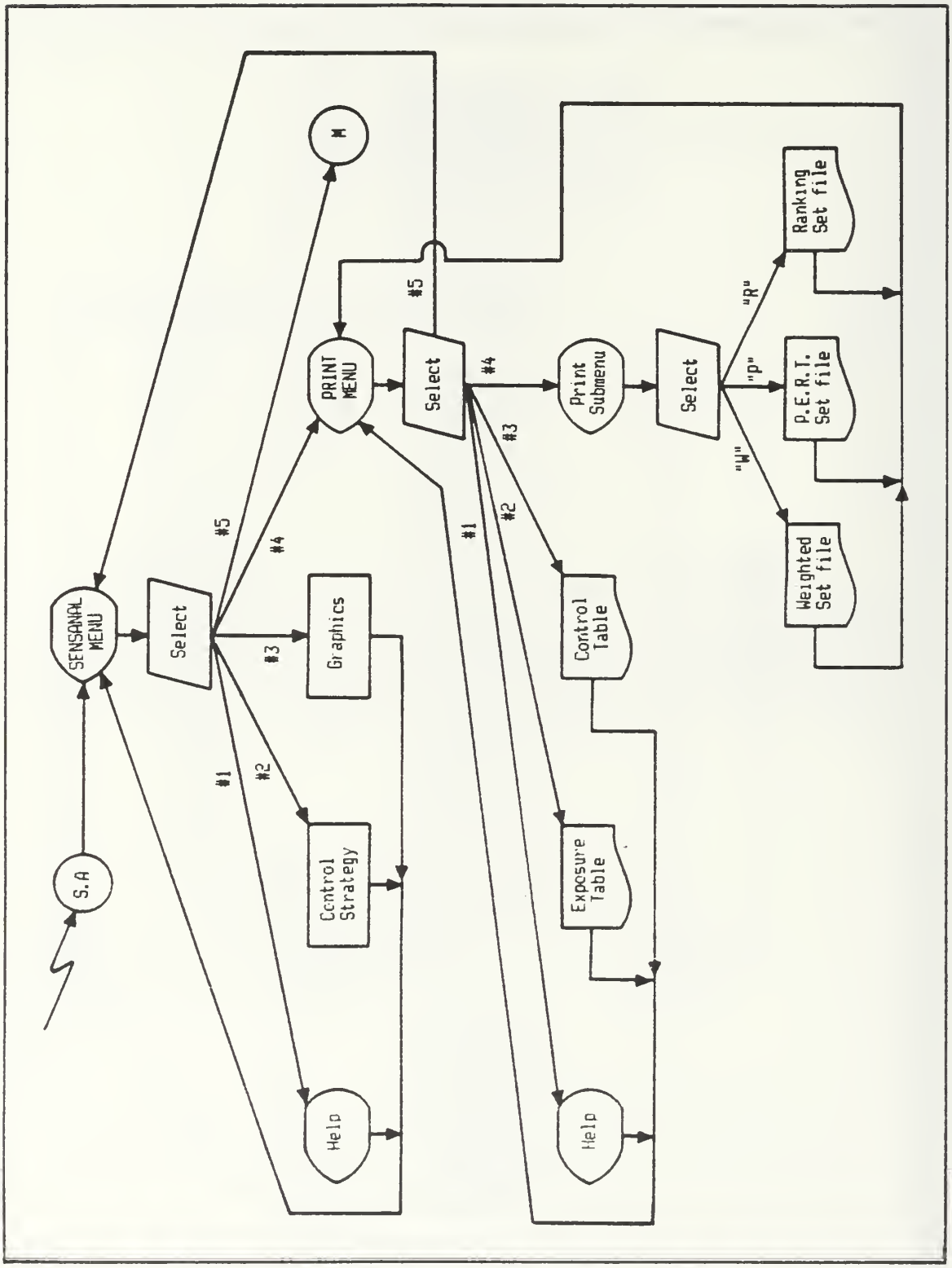


Figure 8.7 Sensitivity Analysis Flow Diagram

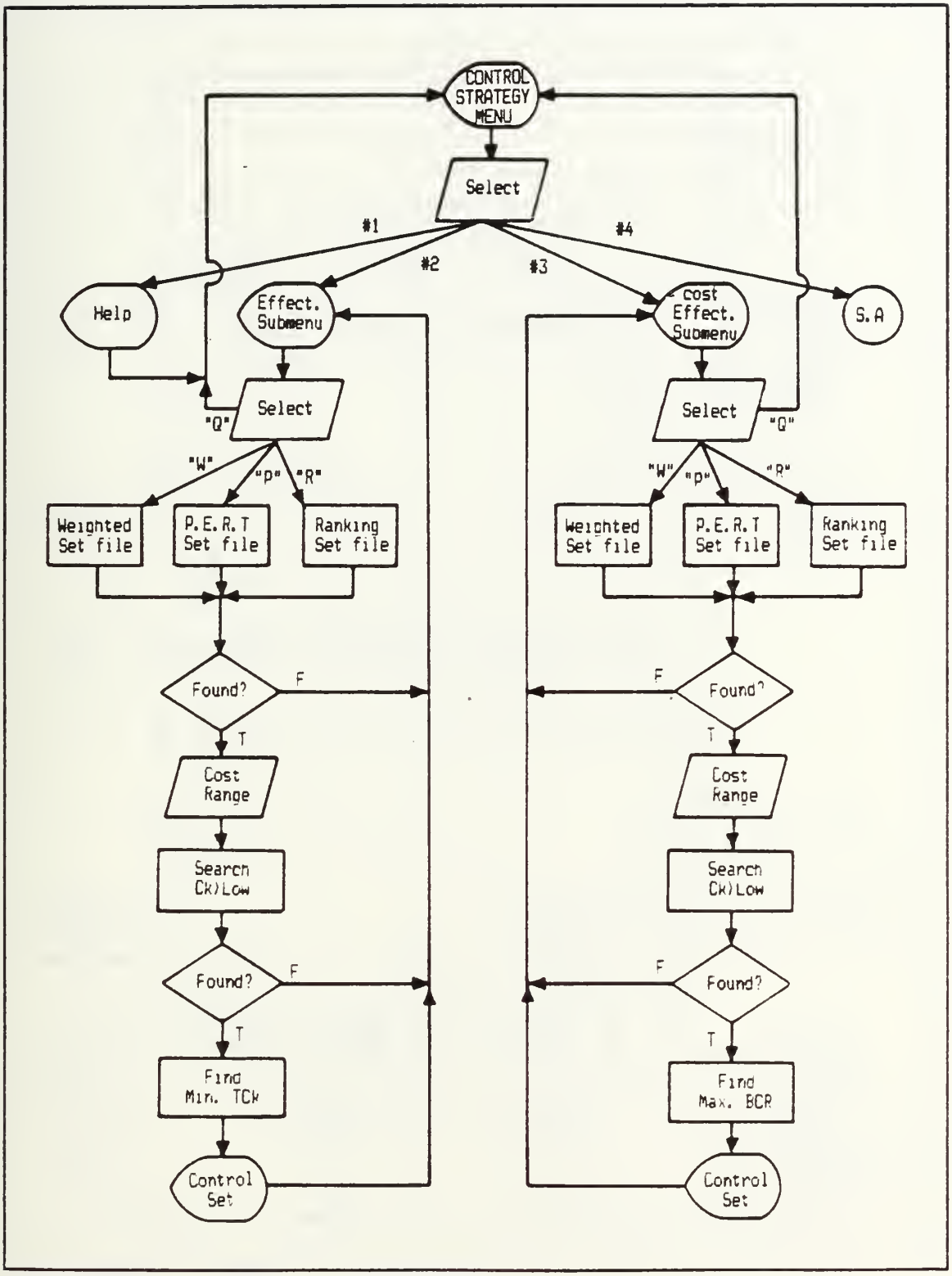


Figure 8.8 Control Strategy Flow Diagram (Sens. Analysis)

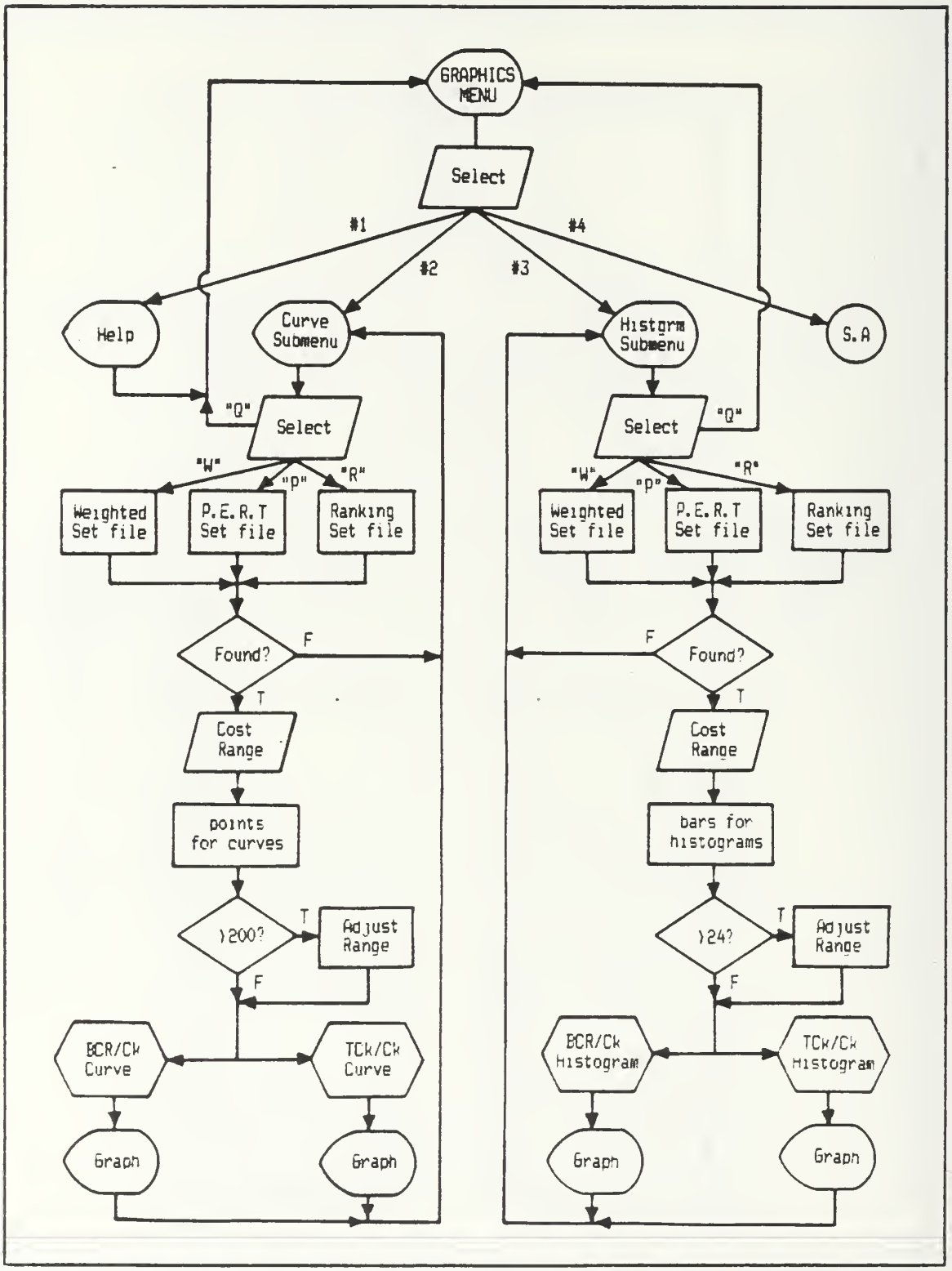


Figure 8.9 Graphics Flow Diagram (Sens. Analysis)

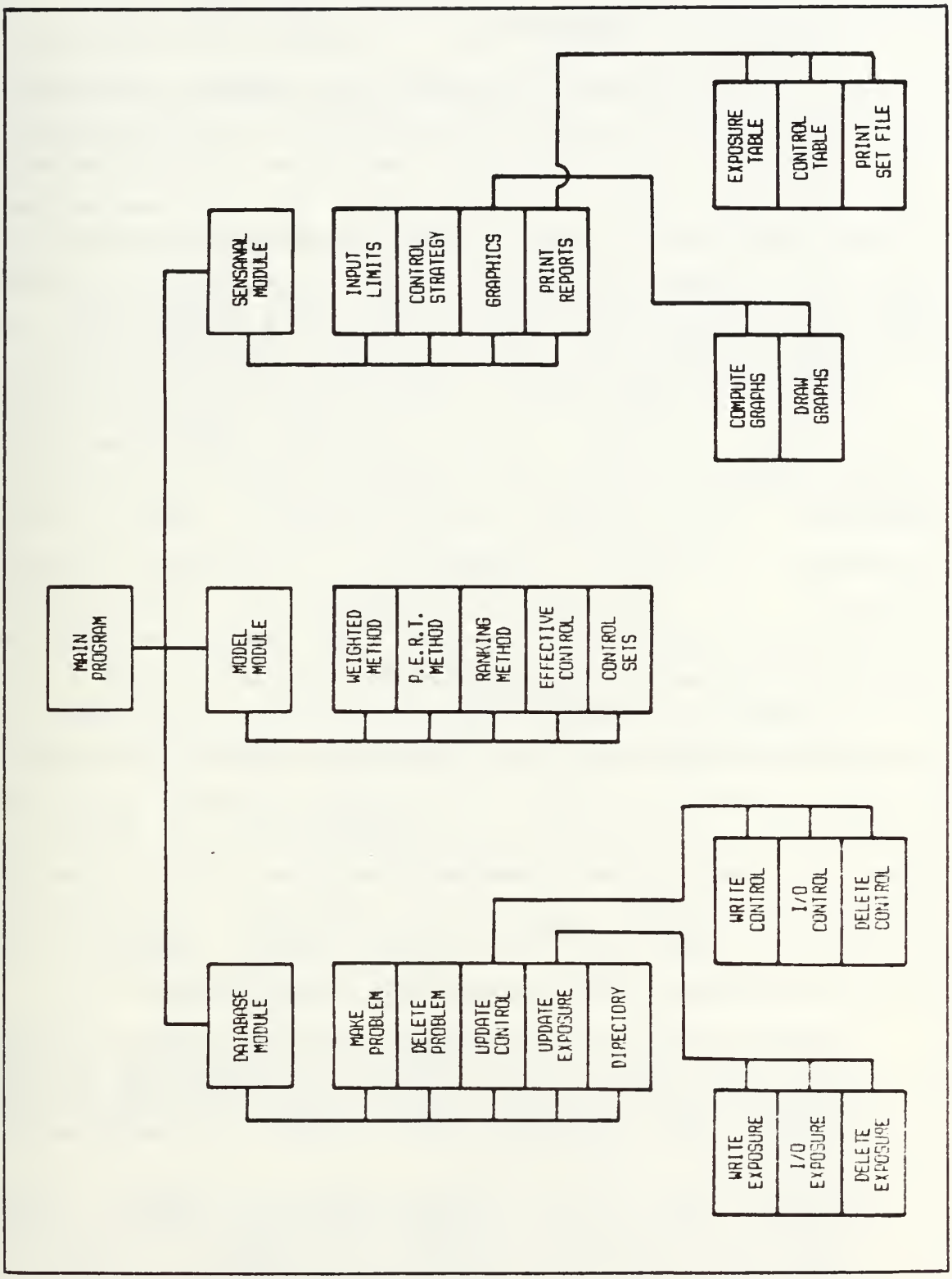


Figure 8.10 The Refined Software Structure

E. IMPLEMENTATION PROBLEMS

The most serious implementation problems are problems associated with the memory management of a microcomputer. The capacity of the memory dedicated to the Central Processing Unit (CPU) is 64 Kbytes for a microcomputer. Consequently, the size of the program part called, along with all the type declarations and the supporting modules, must not exceed the size of the CPU's memory. This is a troublesome limitation when dealing with long programs.

This is the case for the CEA-DSS. The inclusion of the Turbo Access and Turbo Graphix packages within the actual program further limited the allowable size of its modules. Reduction of the module size implies a loose control hierarchy. An effort to reorganize the software structure resulted in undesirable control flow inefficiencies. Fortunately, Turbo Pascal provides for overlay organization which eliminates the memory size limitation.

A technique, called overlays, is used to allow the system to be larger than the amount of memory allocated to it. The idea of overlays is to keep in memory only those instructions and data that are needed at any given time [Ref. 11]. When other instructions are needed, they are loader into space that was previously occupied by instructions that are no longer needed. However, this technique suffers from the following limitations:

- A module must first be loaded into the memory in order to be executed. This causes the system to run somewhat more slowly, due to the extra I/O operation to read the module. For this reason, it is recommended to load the CEA-DSS software on a hard disk or a ram disk. High access speed devices would result in considerable reduction of access time.
- Since overlays share the same space in memory, a module cannot call modules which belong in another overlay of the same area. For example, a module calls another one from a different overlay. This overlay is loaded in place of the caller and the called module is executed. The problem is that after its execution the system is meshed because it does not find the caller to return.

This introduced additional problems to the original structure of the program. The inclusion of control code, like flags, labels, case and if then else statements, helped in establishing communications among the various overlays of the same area.

F. EFFORT DISTRIBUTION FOR THE CEA-DSS DEVELOPMENT

CEA-DSS was built in five months and required an effort of six man-months. Table 6 shows the distribution of the effort, in percentages, among the different phases of the CEA-DSS development.

TABLE 6
EFFORT DISTRIBUTION

Time	Activities
20%	Requirements Analysis and Initial Design
23%	Detailed Design
30%	Programming, Debugging and Testing
5%	Initial Testing and User's feedback
22%	Stepwise refinement of the components

IX. A SESSION WITH THE CEA-DSS

The objective of this chapter is to illustrate the operation of the CEA-DSS. The figures in this chapter have been generated during the testing phase of the CEA-DSS on a IBM PC-XT microcomputer.

A series of screens has been suggested as the most effective way to describe step-by-step the system's basic operation.

STEP 1: Drive definition (Figure 9.1). The system has the flexibility to use a different drive for its database.

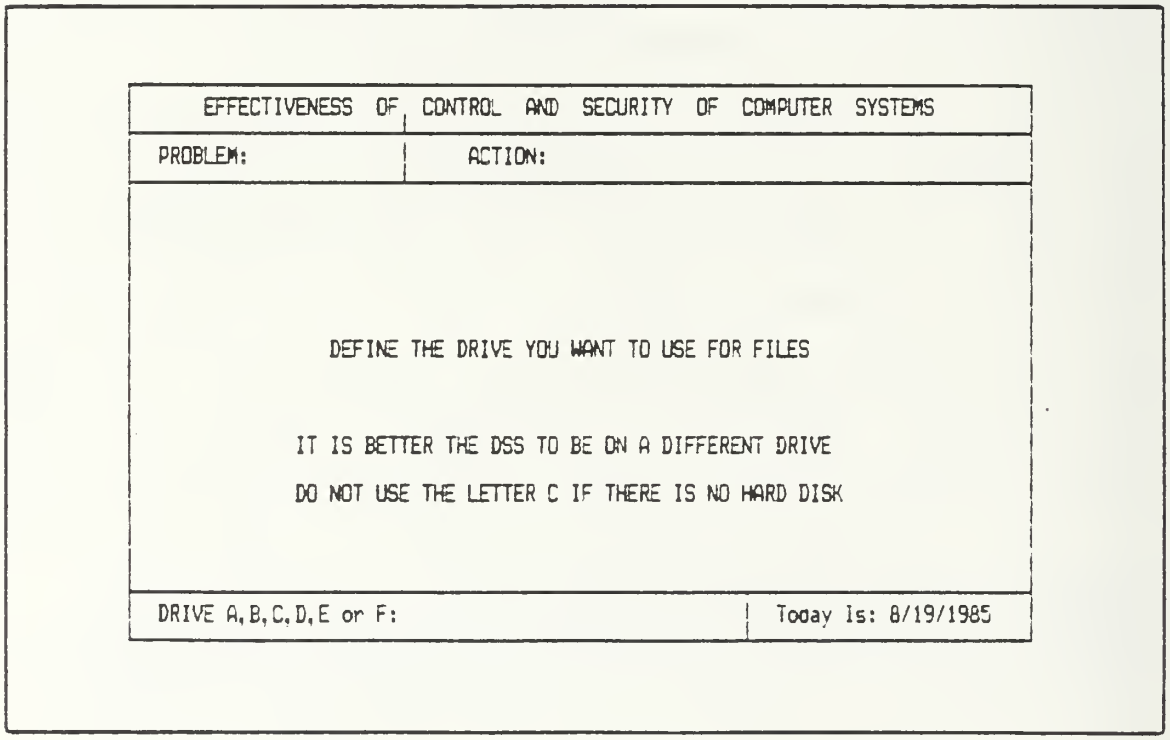


Figure 9.1 Drive Definition

Care must be taken for not using drive "C" with IBM PC-XTs which do not have a hard disk drive. In all other cases,

CEA-DSS has the ability to find any wrong drive definition and prompts the user to redefine the drive.

STEP 2: Directory (Figure 9.2). The directory contains information about previously defined problems. It is located on the drive where data for these problems are stored.

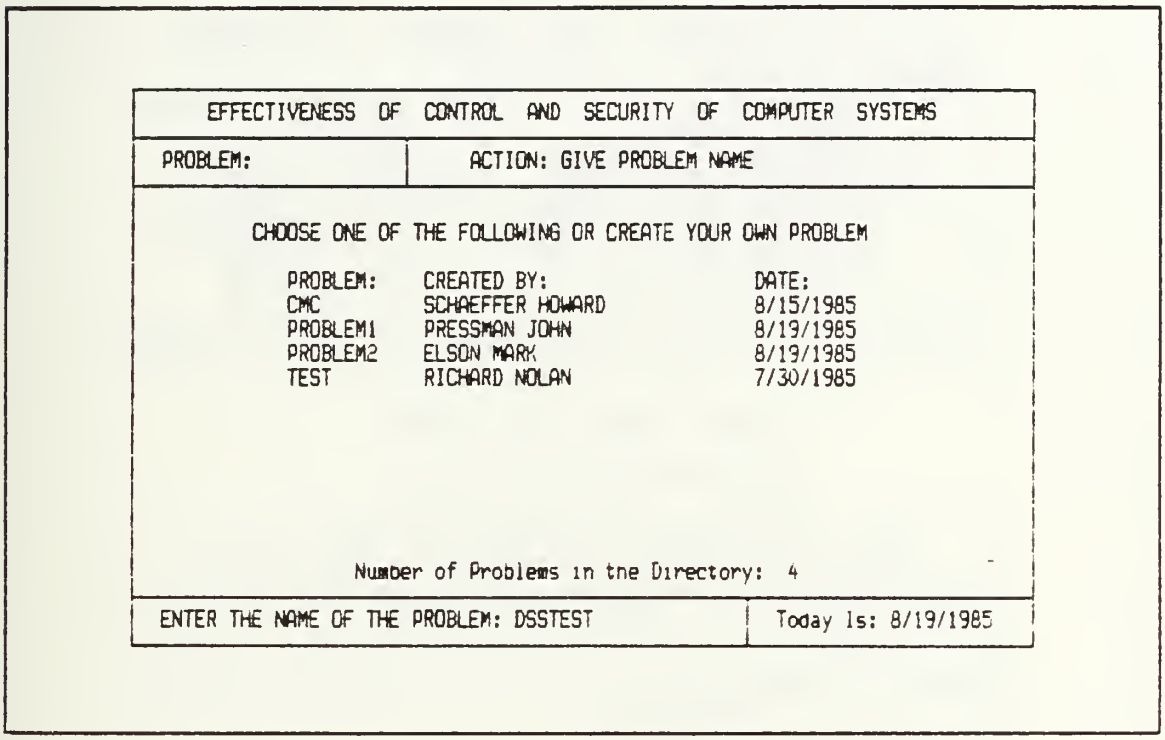
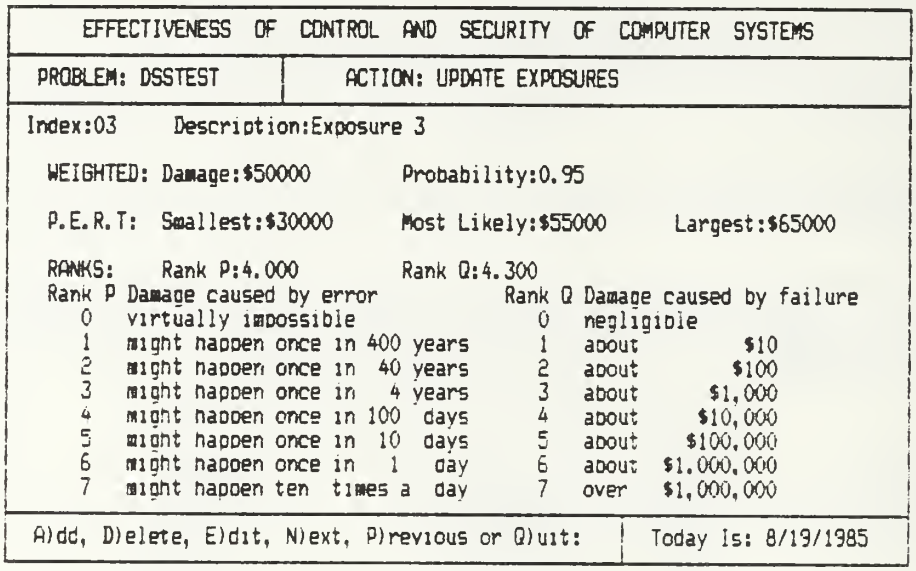


Figure 9.2 Directory

When a new drive, i.e. a new floppy disk, is selected, the system creates a directory first, and then prompts the user to define the problem. For a pre-defined drive, a listing of the directory appears on the frame. The user may select a problem from the directory, or define a new one. In case of an existing problem selection, the process continues with Step 4.

STEP 3: Data entry (Figure 9.3). The system creates the control and exposure files for the particular problem. Then, the user has to provide the initial data. At least



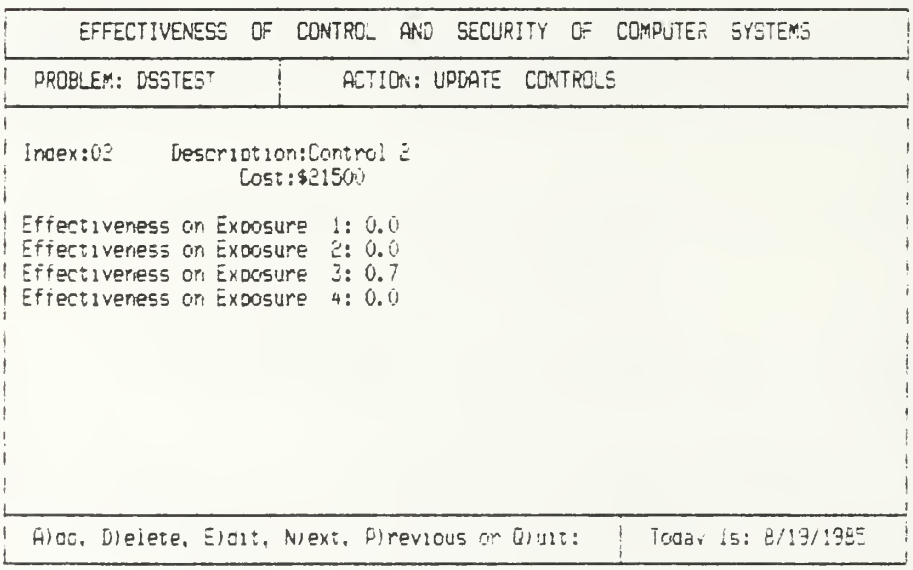


Figure 9.3 Data Entry

two exposures and two control activities are required to enable the CEA-DSS to generate control combinations. The process during this step is under the direct control of the CEA-DSS.

STEP 4: The Main Menu is shown in Figure 9.4. The logical selection for a new problem is the Model option.

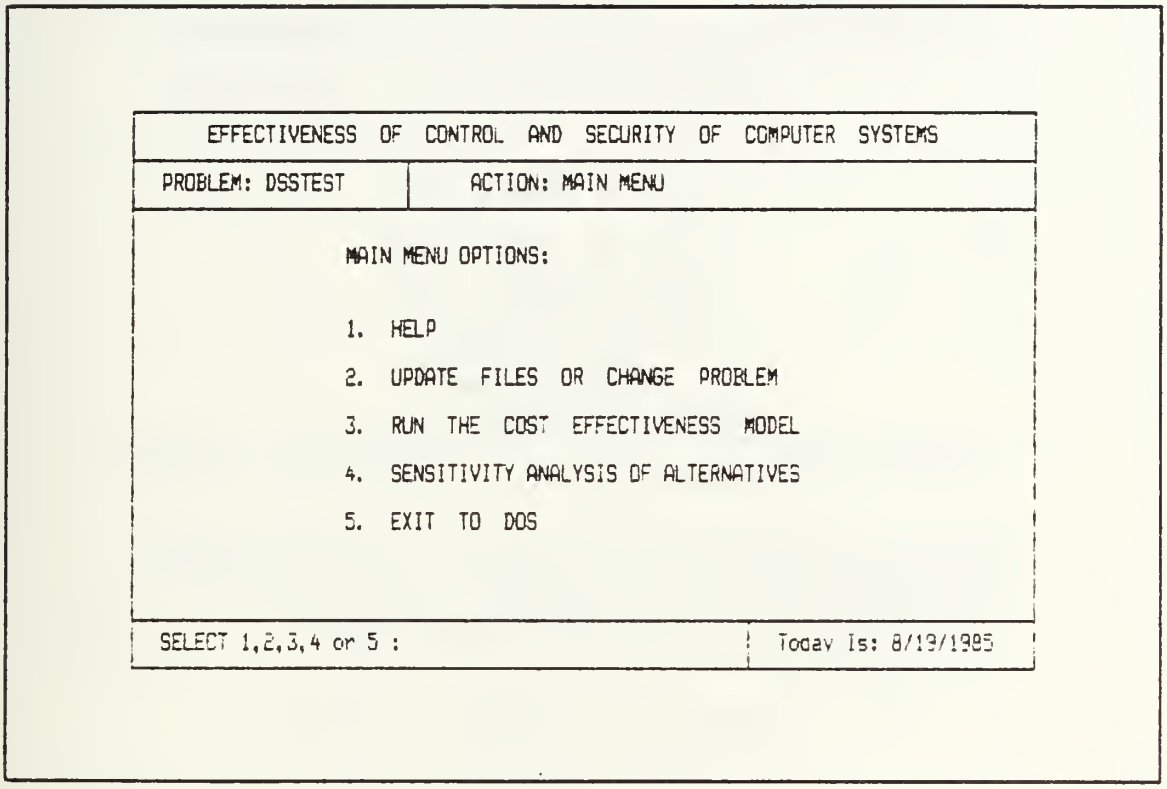


Figure 9.4 Main Menu

STEP 5: Model execution. The Model Menu, allows the selective invocation of one of the three statistical methods for a model run. The user may select one method or all of them. Then, the system prompts the user to define the desired level of cost according to which the generation of control sets will be performed. The use of realistic cost levels is recommended, since it may result in a considerable reduction of the amount of control sets to be generated

and, consequently, in storage and I/O time. Figure 9.5 shows the model menu and the cost level entry.

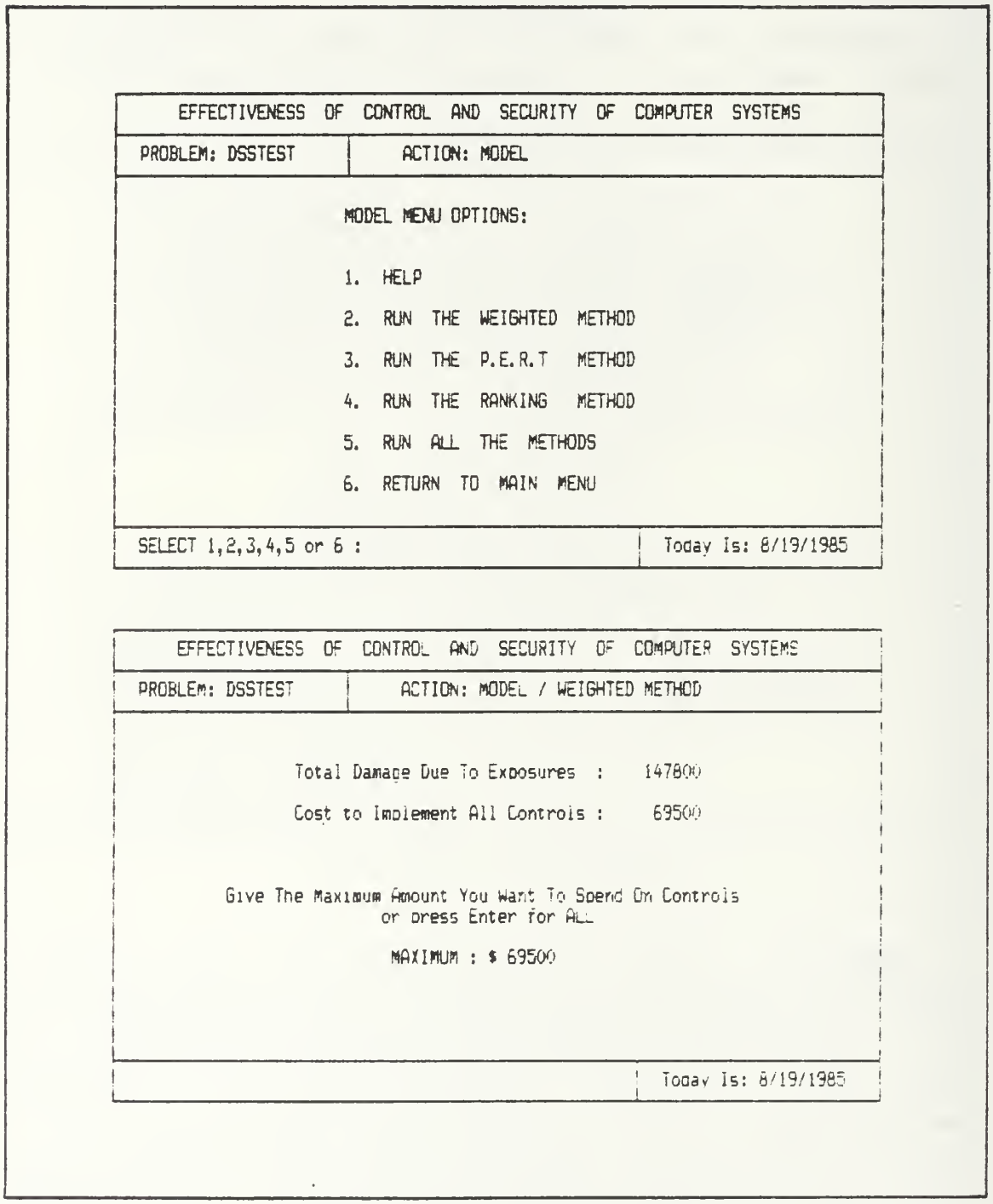


Figure 9.5 Model Menu and Cost Level Entry

STEP 6: Sensitivity Analysis Menu (Figure 9.6). The prerequisite for accessing the "Control Strategy" and the "Graphics" options, is the execution of the model. The same is true and for the "Print Reports" option when a printout of a set file is requested.

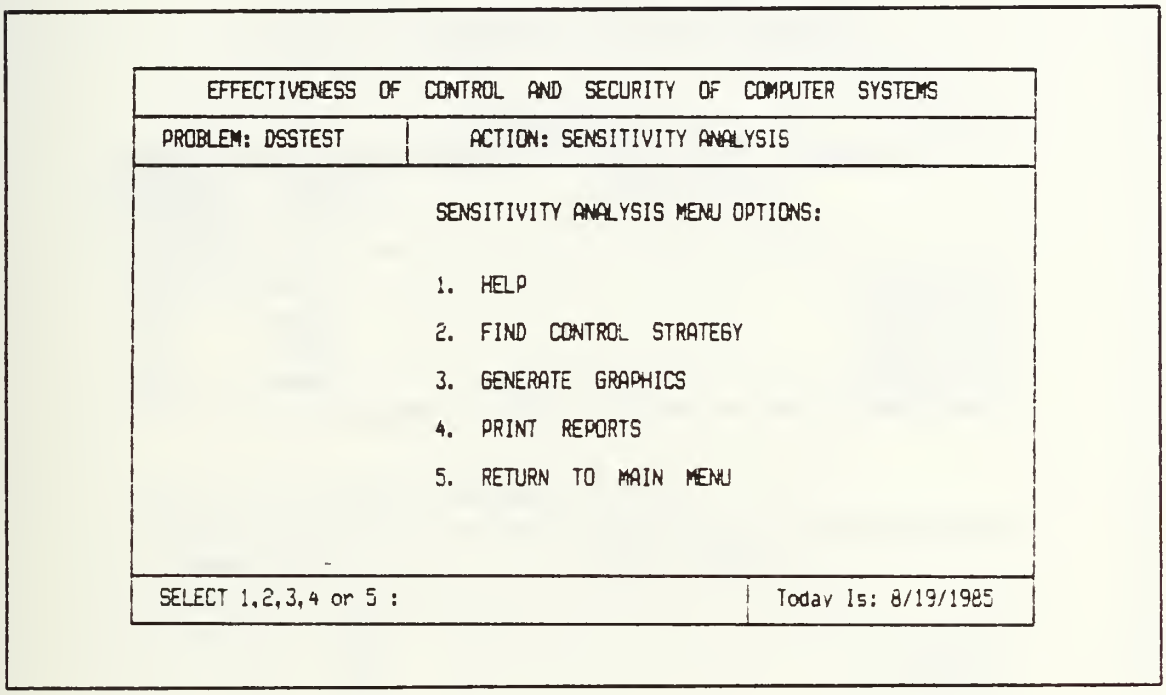


Figure 9.6 Sensitivity Analysis Menu

STEP 7: Print Reports. The system has the capability to produce three types of reports. It is expected that the user will use these reports, during the sensitivity analysis process, as reference. The first table (Figure 9.7) summarizes the initial data of expected losses caused by exposures, for three statistical methods. The second report (Figure 9.8) summarizes the control activities' effectiveness on exposures. Finally, control sets report is a listing of the file created and updated by a model run. Figure 9.9 shows the control sets generated according to the weighted method.

D E C I S I O N S U P P O R T S Y S T E M

COST EFFECTIVENESS ANALYSIS
FOR
CONTROL & SECURITY OF COMPUTER SYSTEMS.

EXPECTED LOSSES CAUSED BY EXPOSURES FOR WORK DSSTEST

THE WEIGHTED METHOD

POTENTIAL ERRORS	AMOUNT OF DAMAGE	PROB/TY OF OCCURENCE
01 Exposure 1	40000	0.850
02 Exposure 2	60000	0.780
03 Exposure 3	50000	0.950
04 Exposure 4	30000	0.650

THE P.E.R.T METHOD

POTENTIAL ERRORS	AMOUNT OF DAMAGE		
	smallest	m.likely	largest
01 Exposure 1	30000	35000	40000
02 Exposure 2	25000	45000	63200
03 Exposure 3	30000	55000	65000
04 Exposure 4	15000	20000	40000

THE RANKING METHOD

POTENTIAL ERRORS	ESTIMATION OF PROBABILITY OF OCCURENCE AND DAMAGE	
	Rank P	Rank G
01 Exposure 1	3.800	4.200
02 Exposure 2	3.850	4.500
03 Exposure 3	4.000	4.300
04 Exposure 4	3.200	4.300

Figure 9.7 An Expected Losses Report

D E C I S I O N S U P P O R T S Y S T E M

COST EFFECTIVENESS ANALYSIS
FOR
CONTROL & SECURITY OF COMPUTER SYSTEMS.

CONTROL ACTIVITIES FOR WORK DSSTEST
01 Control 1
02 Control 2
03 Control 3
04 Control 4

EXPOSURES FOR WORK DSSTEST
01 Exposure 1
02 Exposure 2
03 Exposure 3
04 Exposure 4

	EFFECTIVENESS OF CONTROL a(i) ON EXPOSURE e(i)			
EXPOSURE	01	02	03	04
1	0.800	0.000	0.000	0.000
2	0.000	0.000	0.000	0.830
3	0.000	0.700	0.000	0.000
4	0.000	0.000	0.850	0.000
COST a(i)	13000	21500	10000	25000

Figure 9.8 A Control Effectiveness Report

D E C I S I O N S U P P O R T S Y S T E M

COST EFFECTIVENESS ANALYSIS
FOR
CONTROL & SECURITY OF COMPUTER SYSTEMS.

WEIGHTED METHOD: CONTROL SETS FOR WORK DSSTEST

CONTROL ACTIVITIES USED BY THE CONTROL SETS:
01: Control 1
02: Control 2
03: Control 3
04: Control 4

CONTROL ACTIVITIES	VALUE	COST	EXP.COST	BCR
03,	16575	10000	141225	1.65
01,	27200	13000	133600	2.09
02,	33250	21500	136050	1.54
01,03,	43775	23000	127025	1.90
04,	38844	25000	133956	1.55
02,03,	49825	31500	129475	1.58
01,02,	60450	34500	121850	1.75
03,04,	55419	35000	127381	1.58
01,04,	66044	38000	119756	1.73
01,02,03,	77025	44500	115275	1.73
02,04,	72094	46500	122206	1.55
01,03,04,	82619	48000	113181	1.72
02,03,04,	88669	56500	115631	1.56
01,02,04,	99294	59500	108006	1.66
01,02,03,04,	115869	69500	101431	1.66

Figure 9.9 A Control Sets Report

The print menu is described in Figure 9.10. Before selecting an option, the user must make sure that the printer is on-line.

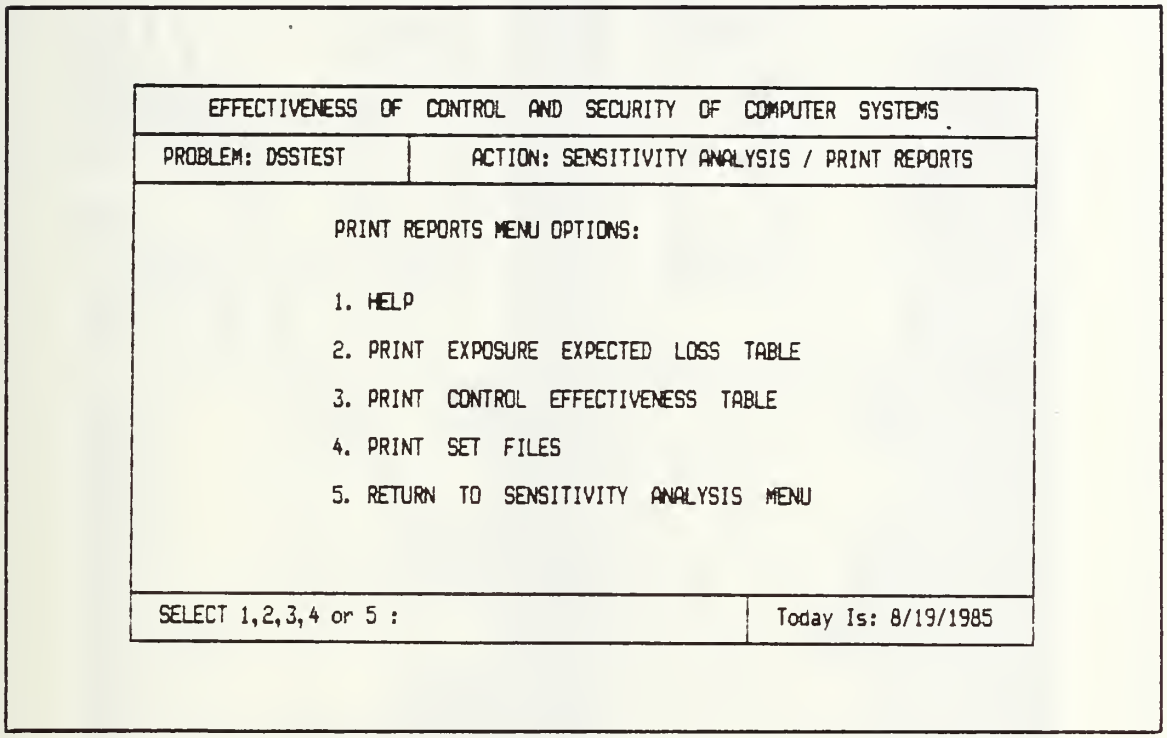


Figure 9.10 The Print Menu

STEP 8: Graphics. Curves and histograms help the user conceptualize the differences among alternative control sets and among different statistical methods. The incompatibility problem of the various types of printers does not allow the system to make hard copies of the graphs. The user can use instead the [PrtSc] key of the keyboard. Each graphics screen contains two graphs. The upper graph depicts the Benefit Cost Ratio versus Cost relationship, and the lower graph the Total Expected Cost versus Cost. Figure 9.11 shows the curves for the DSSTEST problem and Figure 9.12 the histograms. For readability purposes, on each curve can be drawn up to 200 points and on each histogram up to 24 bars.

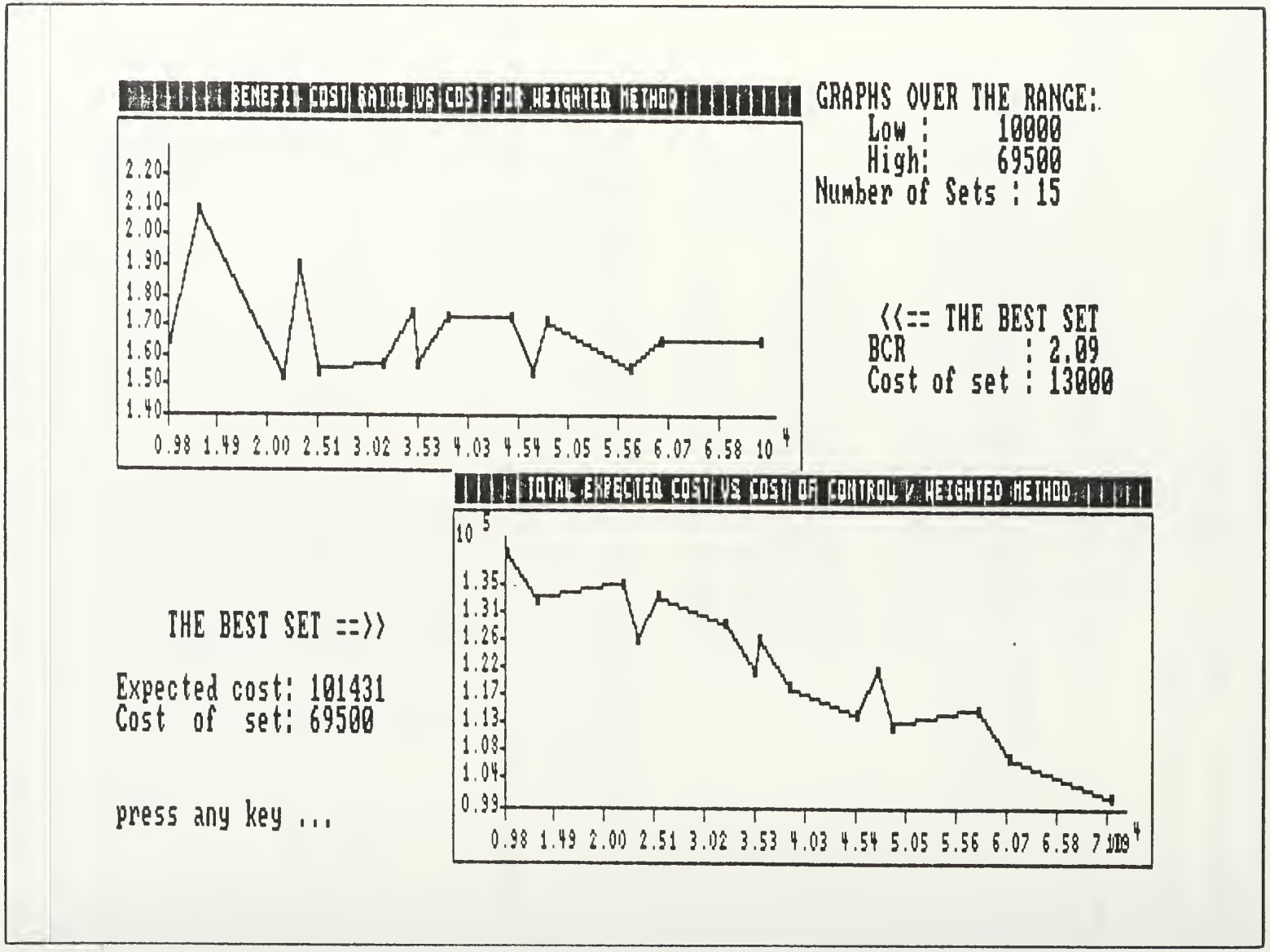


Figure 9.11 Graphical Analysis using Curves

Figure 9.12 Graphical Analysis using Histograms

STEP 9: The last phase of the CEA-DSS process is the control strategy selection. The decision maker may select the most effective (Figure 9.13) or the most cost effective control strategy (Figure 9.14) within the cost range he/she desires. The decision maker, helped by the reports and graphs, is expected to have a better opinion about the amount to be spent for control measures.

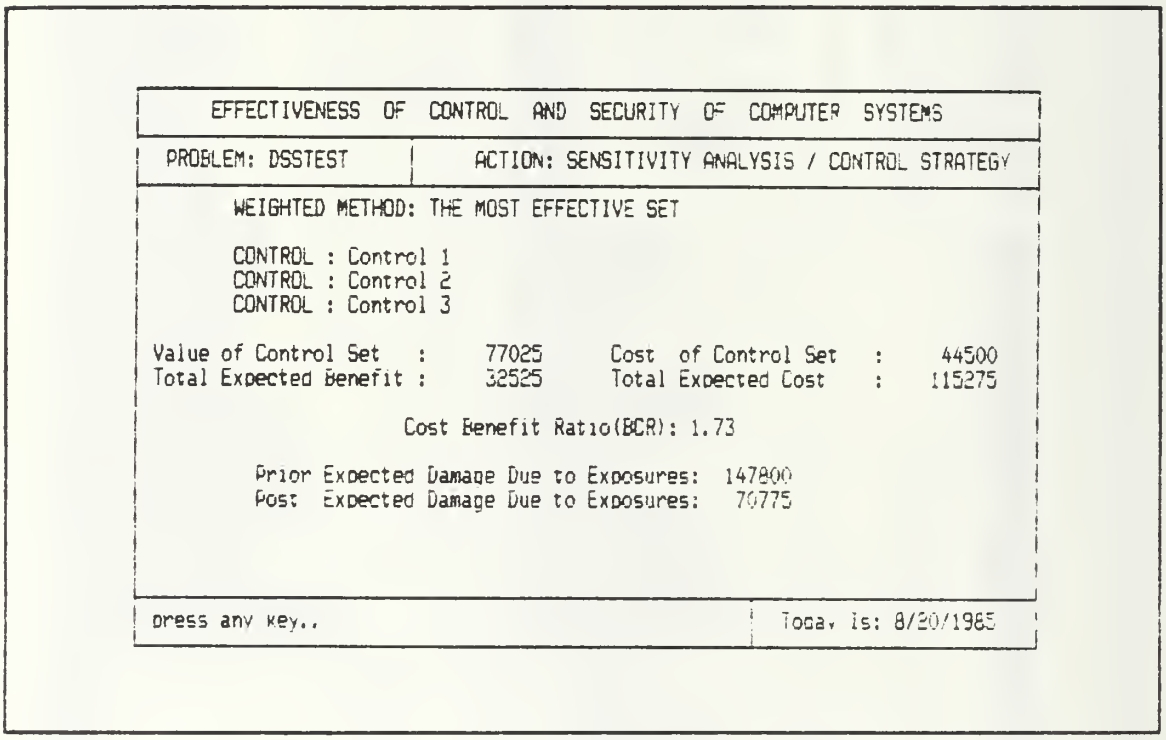


Figure 9.13 The most Effective Control Strategy

The optimal solution in the problem is found when the selected control set is both the most effective and the most cost-effective over a predefined cost range. This is the case for this particular example. Figures 9.13 and 9.14 show the same control set. Under the "Most Effective" option, the control set with the lowest expected cost is selected. Under the "Most Cost-Effective" option, the set with the highest BCR is the most preferable. However,

the system does not provide any algorithm for combining these two options in order to indicate the optimal control strategy.

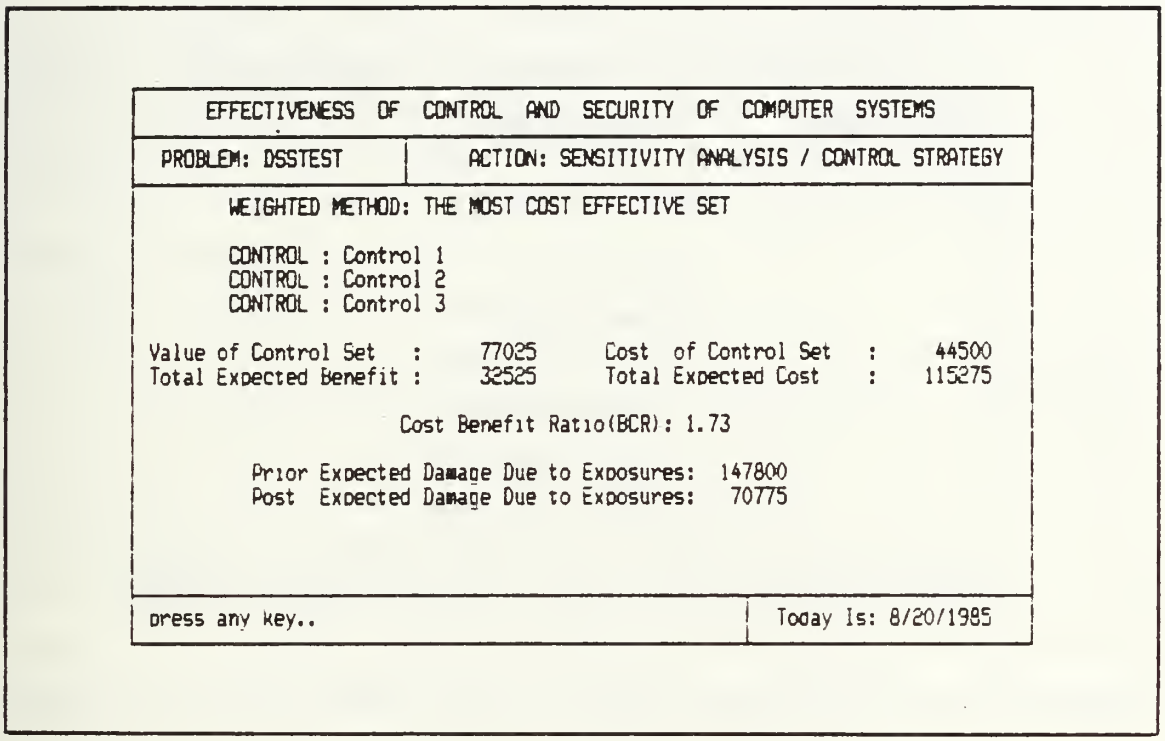


Figure 9.14 The most Cost-Effective Control Strategy

This is the basic process for a problem creation and analysis. Also, the user has the opportunity to access the database of the system through the Database Menu. He/she may modify the initial data, change problem and/or delete the problem. If modifications take place on the data, the model must be executed again. The deletion of the problem erases any file belonging to this as well as its record in the directory. After that, the main menu appears on the screen allowing the user to select one of the "Help", "Database" and "Exit to DOS" options. The other options of the main menu are prohibited when there is no problem definition. The database choice after a problem deletion or changing the

current problem cause the previously described process to be repeated from the beginning. The database menu appears on Figure 9.15.

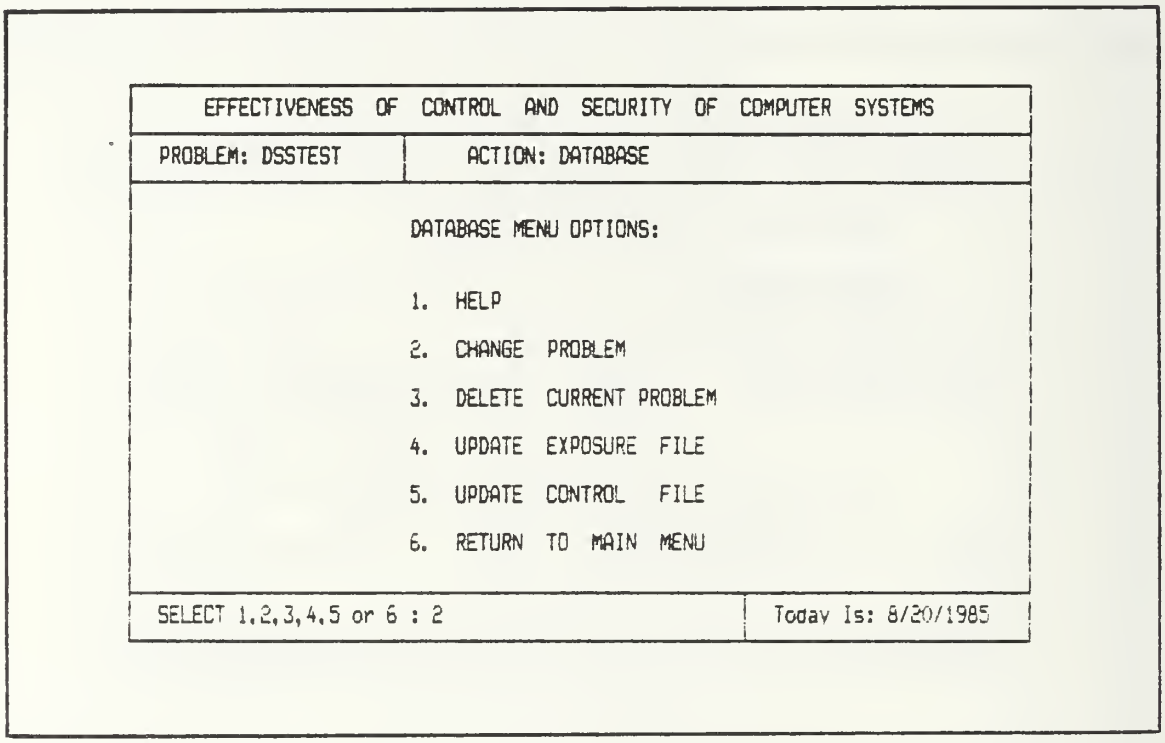


Figure 9.15 Database Menu

X. CONCLUSION

The purpose of the research was to implement a DSS for selecting EDP control strategies. Three analytical methods for determining cost-effectiveness of EDP controls were integrated in a customized database management system. Also a careful user interface was designed to support user interactiveness with the system.

From the user's perspective, the current version of the CEA-DSS is able to handle any uneven condition associated with data entry and process request errors. The enhancement of an acceptable combination of colors and sounds contributes to the user friendliness of the system. Since the users have different preferences, one possible improvement should be to let the user define the colors and sounds he/she likes. The help facility also can be easily modified to satisfy the user needs for on-line information, as discussed in Appendix B.

From the system design perspective, CEA-DSS permits the user to exercise virtual control over its processes. The database system is exclusively designed and implemented to serve the introduced EXPOSURE, CONTROL and SET records. It is expected that any future enhancements in the database schema will require extensive modifications and maintenance to be done on the database and the DBMS. The model base of the CEA-DSS consists of the three variances of the CEA model. Integration of new techniques, using the existing data structure, will require slight modifications of the current system. The same is true for the sensitivity analysis part where any additional reports, graphs and control strategy selection algorithms will not influence the system.

One great inefficiency recorded during the testing phase is associated with the control strategy selection algorithm. The sample problem DSSTEST, presented in Chapter 9, can be considered as an extreme condition. The solution was recognized as the optimal one because it had both, the greatest Benefit Cost Ratio and the lowest Total Expected Cost. This is generally not the case. In reality, the optimal solution is found somewhere in the three dimensional spectrum composed of the Benefit Cost Ratio, the Total Expected Cost and a Scaling Relational Algorithm for them. It is expected that the enhancement of such an algorithm will dramatically improve the control strategy selection process.

Another unresolved issue concerns the assignment of the BCR to the control sets. Control Sets consisting of fewer Control activities turn out to have higher BCRs. This is due to the nature of the algorithm that the model uses to compute the value of the control sets. A way to handle this would be to introduce in the computation of the control set's BCR one more parameter which will be able resolve these differences.

It is recommended that the CEA-DSS built during this research be evaluated on real life applications. In effect, all data used during the testing phase of the CEA-DSS were chosen on a random basis. Information gathered from a real life computer audit process would probably contribute to the evaluation of current control techniques. Furthermore, the CEA-DSS will not only support the selection phase but also the evaluation and exploration phases of the computer audit process life cycle.

APPENDIX A

MESSAGES

INFORMAL MESSAGES:

CREATING EXPOSURE AND CONTROL FILES
Initiation: A new problem has been introduced by the user.

DELETING CONTROL
DELETING EXPOSURE
Initiation: Request for deletion of a control or exposure record.

NEW DIRECTORY
Initiation: New drive definition.

CONTROL "description" IS NOT EFFECTIVE
Initiation: The cost of a control activity is greater than its expected value.

PLEASE WAIT
Initiation: Control sets generation.

PLEASE WAIT FOR THE PREPARATION OF THE GRAPH
Initiation: Request for graphic representation, curve or histogram.

ERROR MESSAGES:

SYSTEM REQUIRES 2 CONTROLS AT LEAST
SYSTEM REQUIRES 2 EXPOSURES AT LEAST
Initiation: Request for deletion of control or exposure record while the file contain only two records.

THE SYSTEM CANNOT HOLD ANOTHER EXPOSURE
Initiation: The user attempts to add the 14th control activity or the 25th exposure.

YOU MUST RUN THE MODEL FIRST
Initiation: Request to access sensitivity analysis areas prior to the model execution.

THERE IS NOT ENOUGH SPACE ON DRIVE X
Initiation: Nonexistent drive definition or the defined drive does not have the appropriate space for a dictionary and problem creation.

CHECK YOUR ENTRY. "HIGH" MUST BE GREATER THAN "LOW"
Initiation: Entry of an ambiguous cost range for the sensitivity analysis part.

TRIGGER MESSAGES:

DEFINE THE DRIVE YOU WANT TO USE FOR FILES
Initiation: CEA-DSS activation or request to change the current problem.

DO YOU WISH TO DELETE THE PROBLEM ?
Initiation: Request to delete the current problem. The system prompts the user to confirm.

THERE IS ALREADY FILE FOR THAT METHOD

Initiation: Request to rerun the model. The user can delete the set file only, or to proceed to model execution.

CANNOT COMPUTE SETS WITHOUT EFFECTIVE CONTROLS

CANNOT COMPUTE SETS WITH ONLY ONE EFFECTIVE CONTROL

Initiation: Control sets generation. The user may correct some initial data or to abandon the current problem.

THERE IS NO ANY SET WITHIN THAT RANGE

Initiation: The cost range defined for sensitivity analysis is very limited. The user may widen the range.

CANNOT MAKE GRAPH WITH LESS THAN 2 SETS

Initiation: Request for graphics, while the defined cost range includes only one control set. The user may redefine a wider cost range.

APPENDIX B

THE HELP FACILITY

The help facility of the CEA-DSS is carried out via the HELP module, listed at the end of the program listing in Appendix C. The Help module supplies the calling menu with information relative to its functions. The algorithm for this selective retrieval of information is based on a code character. This character is sent to the Help module as parameter in the call statement, identifying the calling part of the program. The Help module uses this character to assemble the file name of the text file where the requested information resides.

The advantage of keeping the help text external to the program is that it can be changed easily, with any editor, without affecting the code of the CEA-DSS. The help module also has the advantage of returning control to the caller immediately in case that the requested help, text file, is missing. The files of the system dedicated to the help facility along with their content are listed below.

File: HELPM.TXT

H E L P F O R M A I N M E N U

UPDATE FILES OR CHANGE WORK

This is the Database of the system. You have access to three files. The EXPOSURE, the CONTROL and the PROBLEM file. You can Add, Delete or Edit EXPOSURES and CONTROLS. You can also Change or Delete WORK.

RUN THE COST EFFECTIVENESS MODEL

Once you have updated the EXPOSURE and CONTROL files you can run the model. The model will create the set files

which will be used after for decision making. If you chose the current work from the directory of the system and you are not going to modify the EXPOSURE and CONTROL files you DON'T need to run the model.

SENSITIVITY ANALYSIS

This is the main area of interest. It will help you to find out the optimal solution according to your preferences and budget. There are available to you graphics and print facilities.

File: HELPD.TXT

H E L P F O R D A T A B A S E M E N U

At the bottom of the frame it appears always the command line which prompts you to make selections by typing numbers or letters.

CHANGE PROBLEM

The directory of the system is listed and then you are prompted to define the problem you desire. If you choose an existing one, you will be switched to that immediately. If you create a new one, you will be asked to enter, at least, two EXPOSURES and two CONTROL ACTIVITIES.

DELETE CURRENT PROBLEM

You can only delete the current problem. If you wish to delete a different problem, you must change the problem first, and then choose from the directory the problem you want to delete, and delete it. You will be asked to confirm for the requested deletion by typing the character "!".

UPDATING EXPOSURE OR CONTROL FILE

You can A)dd, D)elete, E)dit Exposures and Controls, and scroll the files forwards and backwards using N)ext or P)revious.

Keep in mind that the edit mode is always in the INSERT MODE.

File: HELPO.TXT

H E L P M O D E L

You can run the model using the WEIGHTED PROBABILITY, the P.E.R.T method and the RANKING method. You will be prompted to enter the upper cost limit. If you have enough controls in the control file it is better to use as short cost ranges as possible in order to minimize the time that the system will require to generate the control sets. Don't forget that N controls may produce 2 to the Nth power control sets.

If you get a message like 'NOT ENOUGH SPACE ON DRIVE X', you can overcome that using one of the following:

1. If you have already run another method for that problem, choose that method again, and erase its set file.
2. Change problem, choose one from the directory that you do not need, delete it, and then choose again the problem you want to work on.

File: HELPS.TXT

H E L P F O R S E N S I T I V I T Y A N A L Y S I S

CONTROL STRATEGY

Control strategy helps you determine the optimal control alternative from all the possible combinations of control activities, or the best one, according to the cost range you are asked to specify.

GRAPHICS

You can generate curves and histograms representing the relations between BENEFIT COST RATIO and COST, or between TOTAL EXPECTED COST and COST.

REPORTS

You can have a hardcopy of the exposures or controls in tabular format, and a listing of the set files.

File: HELPB.TXT

H E L P F O R C O N T R O L S T R A T E G Y

MOST EFFECTIVE ALTERNATIVE

The most effective alternative is the one that it is expected to minimize the total expected cost.

MOST COST EFFECTIVE ALTERNATIVE

The most cost effective alternative is the one that will return the highest benefit per dollar spent.

File: HELPG.TXT

H E L P F O R G R A P H I C S

You can print the curves or histograms by using the [PrtSc] key. Be sure that your printer is ON. The system will switch it to the graphics mode.

Curves and Histograms represent relations of Cost versus Benefit Cost Ratio and Cost versus Total Expected Cost.

Each curve can hold up to 200 control sets to be drawn, and each histogram ut to 24.

You will be asked to give the Cost Range over which the graph will be done. If the number of control sets within the selected range exceeds the above limits, the system will adjust the range.

File: HELPP.TXT

H E L P F O R P R I N T R E P O R T S

YOUR PRINTER MUST BE ON-LINE BEFORE YOU TRY TO PRINT ANY REPORT

You must have set the top of form properly and use page length 11 inches in order the reports to be printed correctly.

APPENDIX C

THE PROGRAM LISTING

```
(*****************************************************************)
(*                                                               *)
(*                    DECISION SUPPORT SYSTEM                    *)
(*                                                               *)
(*                A COST-EFFECTIVENESS ANALYSIS                  *)
(*                             FOR                               *)
(*          CONTROL AND SECURITY OF COMPUTER SYSTEMS             *)
(*                                                               *)
(*                          FILE  DSS.PAS                        *)
(*                                                               *)
(*****************************************************************)

PROGRAM CEA-DSS;
{$A+,I-,R-,V-}

const
  (* TURBO ACCESS CONSTANTS *)
  maxrecsize     = 220;
  maxdatarecsize = maxrecsize;
  maxkeylen      = 11;
  pagesize       = 128;
  order          = 64;
  pagestacksize  = 16;
  maxheight      = 5;

var
  noofrecs : integer;

(* INCLUDE FILES *)
{$IACCESS.BOX}
{$IGETKEY.BOX}
{$IADDKEY.BOX}
{$IDELKEY.BOX}
{$ITYPEDEF.SYS}
{$IGRAPHIX.SYS}
{$IKERNEL.SYS}
{$IWINDOWS.SYS}
{$IHATCH.HGH}
{$ITYPEDEF.DSS}
{$IUTILITY.BOX}
```

```
{$IAXIS.HGH}
{$IPOLYGON.HGH}
{$IHISTOGRM.HGH}        FIRST OVERLAY
{$IMENUS.DSS}
{$IFORMATS.DSS}

{$IHELP.DSS}


{$IDATABASE.DSS}
{$IMODEL.DSS}           SECOND OVERLAY
{$ISENSANAL.DSS}


(* MAIN PROGRAM *)

BEGIN
   textmode;
   textcolor(x);
   textbackground(z);
   help('I');
   ans := ' ';
   makeframe;
   putdate;
   flag := true;
   database;

   while  ans <> '5'  do
   begin
     mainmenu;
     if  flag  then
       select('SELECT 1,2 or 5 : ',['1','2','5'],ans)
     else
       select('SELECT 1,2,3,4 or 5 : ',['1'..'5'],ans);
      case ans of
       '1' : help('M');
       '2' : database;
       '3' : model;
       '4' : sensitivityanalysis
     end {of case}
   end; {of while}

   clrscr;
   gotoxy(15,12);
   write('***** END OF THE DECISION SUPPORT SYSTEM *****');
   wait;
   textcolor(15);
   textbackground(0);
   clrscr
END.
```

```
(*****************************************************************)
(*                                                               *)
(*                    TYPE DECLARATIONS                          *)
(*                                                               *)
(*                    FILE  TYPEDEF.DSS                          *)
(*                                                               *)
(*****************************************************************)

const
  maxctrl = 13; { maximum number of control activities }
  maxexp  = 24; { maximum number of exposures }
  z       = 0;  {standard text background}
  x       = 14; {standard text color}

type
  chset = set of char;

  str2    = string[2];
  str5    = string[5];
  str8    = string[8];
  str10   = string[10];
  str25   = string[25];
  str40   = string[40];
  str50   = string[50];
  str80   = string[80];
  anystr  = string[255];

  ctrlrange = 0..maxctrl;
  exprange  = 0.. maxexp;

  exposure       = record
    index        : str2;
    description  : str50;
    damage       : str8;
    probability  : str5;
    smallest,
    mostlikely,
    largest      : str8;
    rankP,
    rankQ        : str5
  end;

  eff            = array[1..maxexp] of string[5];
  ctrleff        = array[1..maxctrl] of eff;
  control        = record
    index        : str2;
    description  : str50;
    cost         : str8;
    effect       : eff
  end;
```

```
  controlmatrix = array[1..maxctrl] of control;
  combinationindex = array[1..maxctrl] of str2;

  setrec          = record
    setcomb       : combinationindex;
    Vk,Lk,Ck,
    Nk,TCk        : str10;
    BCR           : str5
  end;

  problemrec      =  record
    problemname   :  str8;
    creator       :  str25;
    date          :  str10;
    wcomb,pcomb,
    rcomb         : combinationindex;
    wtotcost,
    ptotcost,
    rtotcost      : str10
  end;


var
  file1, file2    : datafile;
  index1,index2   : indexfile;
  fl              : file;
  expsr           : exposure;
  ctrl            : control;
  st              : setrec;
  problem         : problemrec;
  cproblem        : str8;
  wcombindex,
  pcombindex,
  rcombindex,
  comb            : combinationindex;
  ce              : ctrleff;
  cc              : array[1..maxctrl] of str8;
  ctrlmatrix      : controlmatrix;
  totaloss,
  totalcost,
  wtotalcost,
  ptotalcost,
  rtotalcost      : real;
  expno           : integer;
  ans, tc         : char;
  dr              : str2;
  flag            : boolean;
```

```
(*************************************************************)
(*                                                           *)
(*                      UTILITY.BOX                          *)
(*                                                           *)
(*     The utility box contains all the procedures and       *)
(*     functions  which  are commonly  used by all the       *)
(*     modules of the system.                                *)
(*                                                           *)
(*************************************************************)


(* upcasestr converts a string to upper case *)
function upcasestr(s : str80) : str80;
var
  p : integer;

begin
  for p := 1 to length(s) do
    s[p] := upcase(s[p]);
  upcasestr := s;
end;


(* conststr returns a string with N characters of value C*)
function conststr(c : char; n : integer) : str80;
var
  s : string[80];

begin
  if  n < 0  then
    n := 0;
  s[0] := chr(n);
  fillchar(s[1],n,c);
  conststr := s;
end;


(* getvalue returns the ASCII value of a string *)
function getvalue(s : anystr) : integer;
var
   i, total : integer;

begin
   total := 0;
   if  length(s) > 0  then
      for  i := 1 to length(s) do
         total := total + ord(copy(s,i,1));
   getvalue := total
end;
```

```
(* strtoreal returns a real number equivalent to a string *)
function  strtoreal(s : str10) : real;
var
  t : integer;
  r : real;

begin
  val(s,r,t);
  strtoreal := r
end;


(* realtostr returns a string equivalent to a real number *)
function  realtostr(r : real) : str10;
var
  s : string[10];

begin
  fillchar(s,sizeof(s),0);
  str(r,s);
  realtostr := s
end;


(* strtoint returns an integer equivalent to a string *)
function  strtoint(s : str2) : integer;
var
  i,j : integer;

begin
  val(s,i,j);
  strtoint := i
end;


(* intostr returns a string equivalent to an integer *)
function  inttostr(n : integer) : str2;
var
  s : string[2];

begin
  fillchar(s,2,0);
  str(n,s);
  inttostr := s
end;
```

```
(* adjuststr removes any leadind spaces from a string *)
procedure  adjuststr(var s : anystr);
begin
  while  s[1] = ' '  do
    if  s[1] = ' '  then
      delete(s,1,1);
end;


(* the system waits for the user *)
procedure  wait;
var
  ch  : char;
  i,j : integer;

begin
  textcolor(15);
  gotoxy(2,23); write(conststr(' ',53));
  gotoxy(3,23); write('press any key..');
  for  i := 1 to 3  do
  begin
    j := sqr(random(30))+300;
    sound(j); delay(300)
  end;
  nosound;
  read(kbd,ch);
  gotoxy(3,23); write('                ');
  textcolor(x)
end;


(*  Beep sounds the terminal bell or beeper *)
procedure beep;
begin
  sound(680); delay(400); nosound
end;


(* inputstr is used for the entry and validation of data.
   It enables also the  use of the cursor  movement keys
   char-left, char-right and del. of the keyboard.      *)
procedure inputstr(var s     : anystr;
                       l,i,j : integer;
                       term  : chset;
                   var tc    : char    );
label
  again;
var
  valid : set of char;
  value,
  p, n  : integer;
  ch    : char;
```

```
begin
  textbackground(14);
  textcolor(0);
  tc := #0;
  valid := term + [#8,#13,#27];
  again:
  gotoxy(i,j); write(s,conststr(' ',l-length(s)));
  p := 0;
  repeat
    gotoxy(i+p,j); read(kbd,ch);
    if  not (ch in valid)  then
      beep
    else
    begin
      if  (ch in term) and (p < l)  then
      begin
        p := p + 1;
        delete(s,l,1);
        insert(ch,s,p);
        write(copy(s,p,l))
      end;
      if  (ch = #8) and (p >= length(s)) and (p > 0) then
      begin
        delete(s,p,1);
        p    := p - 1;
        gotoxy(i+p,j); write(' ')
      end;
      if  (ch = #27) and keypressed  then
      begin
        read(kbd,ch);
        if  ch = 'K'  then
        begin
          if  p > 0  then
            p := p - 1
          else
            beep
        end;
        if  (ch = 'M') and (p < length(s))  then
          p := p + 1;
        if  (ch = 'S') and (p < length(s))  then
        begin
          delete(s,p+1,1);
          write(copy(s,p+1,l),' ')
        end;
        if  ch in ['H','P']  then
        begin
          tc := chr(100 + ord(ch));
          p := 1
        end
      end
    end
  until  (ch = #13) or (p = l);
```

```
  if  ch = #13  then
    tc := ch;
  value := getvalue(s);
  n := 32 * length(s);
  if  (value <= n) and (ch <> 'H')  then
  begin
    beep;
    tc := #0;
    goto again
  end;
  adjuststr(s);
  textbackground(z);
  textcolor(x)
end; {inputstr}


(* action writes on the frame the current action *)
procedure action(s : str40);
begin
  textcolor(2);
  gotoxy(39,4); write(conststr(' ',40));
  gotoxy(39,4); write(s);
  textcolor(x)
end;


procedure  clearmessage;
begin
  gotoxy(2,12); write(conststr(' ',78))
end;


(* message writes a string at the center of the frame *)
procedure  message(s : str80);
var
  i : integer;
begin
  clearmessage;
  textbackground(0);
  textcolor(31);
  i := trunc((80 - length(s))/2);
  gotoxy(i,12); write(copy(s,1,length(s)));
  beep;
  textbackground(z);
  textcolor(x)
end;


procedure clearselect;
begin
   gotoxy(2,23); write(conststr(' ',53))
end;
```

```
(* select writes the command line at the bottom of the
   frame and accepts the selection                        *)
procedure select(    prompt : str80;
                     term   : chset;
                 var tc     : char    );
var
  ch : char;

begin
  clearselect;
  textcolor(15);
  gotoxy(4,23);  write(prompt);
  textbackground(30);
  gotoxy(5+length(prompt),23); write(' ');
  gotoxy(5+length(prompt),23);
  textbackground(z);
  textcolor(x);
  repeat
    read(kbd,ch);
    tc := upcase(ch);
    if  not (tc in term)  then
      beep;
  until tc in term;
  write(tc)
end;


(* cleartext clears the work area of the frame *)
procedure cleartext;
var
   i : integer;
begin
   for  i := 10 to 21  do
   begin
      gotoxy(2,i);
      write(conststr(' ',78))
   end
end;


procedure clearframe;
var
   i: integer;
begin
   for  i := 6 to 9  do
   begin
      gotoxy(2,i);
      write(conststr(' ',78))
   end;
   cleartext;
   clearselect;
end;
```

```
procedure  clearproblem;
begin
  gotoxy(13,4); write(conststr(' ',8))
end;


(* problemfield writes the problem description in the
   problem area ofthe frame                              *)
procedure  problemfield(s : str8);
begin
  clearproblem;
  textcolor(2);
  gotoxy(13,4);
  write(s);
  textcolor(x)
end;


(* funckey helps in using the cursor movement keys
   line-up,line down,and enter of the keyboard           *)
procedure funckey(ch : char; var i : integer);
begin
  if  ch > #126  then
    ch := chr(ord(ch)-100);
  if  ch = 'P'  then
      i := i + 1;
  if  ch = 'H'  then
  begin
    if  i = 1  then
      beep
    else
      i := i - 1
  end;
  if  ch = #13  then
      i := i + 1
end;


(* availablespace returns the available space(bytes)
of the    logged drive.                                 *)
procedure spaceavailable( var totalbytes : real );
type
  regrec =  record   { register pack Used in MSDos call }
      AX, BX, CX, DX, BP, SI, DI, DS, ES, Flags : integer;
    end;
var
  tracks,
  drive,
  bytes,
  sectors     : integer;
  regs        : regrec;
  ch          : char;
```

```
procedure diskstatus( drive : integer;  var tracks,
                            bytes, sectors : integer );
begin
  regs.AX := $3600;
  regs.DX := Drive;
  MSDos( regs );
  tracks := regs.BX;
  bytes := regs.CX;
  sectors := regs.AX
end;
begin
  ch := copy(dr,1,1);
  drive := ord(ch) - 64;
  diskstatus( drive, tracks, bytes, sectors );
  totalbytes := (( sectors * bytes * 1.0 ) * tracks )
end;


procedure getdate( var date : str10 );
type
  regrec =  record    { register pack Used in MSDos call }
      AX, BX, CX, DX, BP, SI, DI, DS, ES, Flags : integer;
  end;
var
  regs  : regrec;
  mm,dd : string[2];
  yy    : string[4];
begin
  regs.ax := $2A shl 8;
  msdos(regs);
  str(regs.cx,yy);
  str(regs.dx mod 256,dd);
  str(regs.dx shr 8,mm);
  date := mm+'/'+dd+'/'+yy
end;


(* putdate writes the date at the lower right corner
   of the frame                                        *)
procedure putdate;
var
  date : string[10];
begin
  textbackground(3);
  textcolor(0);
  getdate(date);
  gotoxy(68,23); write(date);
  textbackground(z);
  textcolor(x)
end;
```

```
(*****************************************************************)
(*                                                               *)
(*                          MENUS.DSS                            *)
(*                                                               *)
(*****************************************************************)


overlay procedure mainmenu;
begin
  clearframe;
  action('MAIN MENU');
  gotoxy(20, 7);
  write('MAIN MENU OPTIONS:');
  gotoxy(20,10);
  write('1.  HELP');
  gotoxy(20,12);
  write('2.  UPDATE  FILES  OR  CHANGE  PROBLEM');
  gotoxy(20,14);
  write('3.  RUN  THE  COST  EFFECTIVENESS  MODEL');
  gotoxy(20,16);
  write('4.  SENSITIVITY ANALYSIS OF ALTERNATIVES');
  gotoxy(20,18);
  write('5.  EXIT  TO  DOS');
end;


overlay procedure  dbasemenu;
begin
  clearframe;
  action('DATABASE');
  gotoxy(26, 7);
  write('DATABASE MENU OPTIONS:');
  gotoxy(26,10);
  write('1.  HELP');
  gotoxy(26,12);
  write('2.  CHANGE  PROBLEM');
  gotoxy(26,14);
  write('3.  DELETE  CURRENT PROBLEM');
  gotoxy(26,16);
  write('4.  UPDATE  EXPOSURE  FILE');
  gotoxy(26,18);
  write('5.  UPDATE  CONTROL   FILE');
  gotoxy(26,20);
  write('6.  RETURN  TO  MAIN  MENU');
end;
```

```
overlay procedure  modelmenu;
begin
  clearframe;
  action('MODEL');
  gotoxy(25, 7);
  write('MODEL MENU OPTIONS:');
  gotoxy(25,10);
  write('1.  HELP');
  gotoxy(25,12);
  write('2.  RUN  THE  WEIGHTED  METHOD');
  gotoxy(25,14);
  write('3.  RUN  THE  P.E.R.T   METHOD');
  gotoxy(25,16);
  write('4.  RUN  THE  RANKING   METHOD');
  gotoxy(25,18);
  write('5.  RUN  ALL  THE  METHODS');
  gotoxy(25,20);
  write('6.  RETURN  TO  MAIN  MENU');
end;


overlay procedure  sensanalmenu;
begin
  clearframe;
  action('SENSITIVITY ANALYSIS');
  gotoxy(28, 7);
  write('SENSITIVITY ANALYSIS MENU OPTIONS:');
  gotoxy(28,10);
  write('1.  HELP');
  gotoxy(28,12);
  write('2.  FIND  CONTROL  STRATEGY');
  gotoxy(28,14);
  write('3.  GENERATE  GRAPHICS');
  gotoxy(28,16);
  write('4.  PRINT  REPORTS');
  gotoxy(28,18);
  write('5.  RETURN  TO  MAIN  MENU');
end;
```

```
overlay procedure  controlstrategymenu;
begin
  clearframe;
  action('SENSITIVITY ANALYSIS / CONTROL STRATEGY');
  gotoxy(15, 7);
  write('CONTROL STRATEGY MENU OPTIONS:');
  gotoxy(15,10);
  write('1.  HELP');
  gotoxy(15,13);
  write('2.   FIND   THE   MOST   EFFECTIVE  CONTROL  SET');
  gotoxy(15,16);
  write('3.  FIND   THE   MOST   COST   EFFECTIVE  CONTROL',
                                              ' SET');
  gotoxy(15,19);
  write('4.    RETURN    TO   SENSITIVITY  ANALYSIS  MENU');
end;


overlay procedure  graphicsmenu;
begin
  clearframe;
  action('SENSITIVITY ANALYSIS / GRAPHICS');
  gotoxy(21, 7);
  write('GRAPHICS MENU OPTIONS:');
  gotoxy(21,10);
  write('1.  HELP');
  gotoxy(21,13);
  write('2.  DRAW  REPRESENTATIVE  CURVES');
  gotoxy(21,16);
  write('3.  DRAW  REPRESENTATIVE  HISTOGRAMS');
  gotoxy(21,19);
  write('4.  RETURN TO SENSITIVITY ANALYSIS MENU');
end;


overlay procedure  printmenu;
begin
  clearframe;
  action('SENSITIVITY ANALYSIS / PRINT REPORTS');
  gotoxy(19, 7);
  write('PRINT REPORTS MENU OPTIONS:');
  gotoxy(19,10);
  write('1. HELP');
  gotoxy(19,12);
  write('2. PRINT    EXPOSURE    EXPECTED    LOSS   TABLE');
  gotoxy(19,14);
  write('3. PRINT  CONTROL  EFFECTIVENESS  TABLE');
  gotoxy(19,16);
  write('4. PRINT  SET  FILES');
  gotoxy(19,18);
  write('5.  RETURN    TO    SENSITIVITY   ANALYSIS  MENU');
end;
```

```
(*************************************************************)
(*                                                           *)
(*                  F O R M A T S   D S S                    *)
(*                                                           *)
(*************************************************************)


(* makeframe writes the frame of the system along with
   the identification of each area                          *)
procedure makeframe;
var
   i : integer;
begin
   clrscr;
   gotoxy( 1,1);  write(chr(201));
   gotoxy(80,1);  write(chr(187));
   gotoxy( 2,1);  write(conststr(chr(205),78));
   gotoxy( 1,2);  write(chr(186));
   gotoxy(80,2);  write(chr(186));
   gotoxy( 1,3);  write(chr(204));
   gotoxy(80,3);  write(chr(185));
   gotoxy( 2,3);  write(conststr(chr(205),78));
   gotoxy(25,3);  write(chr(203));
   gotoxy( 1,4);  write(chr(186));
   gotoxy(25,4);  write(chr(186));
   gotoxy(80,4);  write(chr(186));
   gotoxy( 1,5);  write(chr(204));
   gotoxy(80,5);  write(chr(185));
   gotoxy( 2,5);  write(conststr(chr(205),78));
   gotoxy(25,5);  write(chr(202));

   for  i := 6 to 21  do
   begin
     gotoxy( 1,i);  write(chr(186));
     gotoxy(80,i);  write(chr(186))
   end;

   gotoxy( 1,22);  write(chr(204));
   gotoxy(80,22);  write(chr(185));
   gotoxy( 2,22);  write(conststr(chr(205),78));
   gotoxy(55,22);  write(chr(203));
   gotoxy( 1,23);  write(chr(186));
   gotoxy(55,23);  write(chr(186));
   gotoxy(80,23);  write(chr(186));
   gotoxy( 1,24);  write(chr(200));
   gotoxy(80,24);  write(chr(188));
   gotoxy( 2,24);  write(conststr(chr(205),78));
   gotoxy(55,24);  write(chr(202));
   textcolor(3);
```

```
   gotoxy(8,2);
   write('EFFECTIVENESS  OF  CONTROL  AND  SECURITY  OF',
                                      ' COMPUTER  SYSTEMS');
   gotoxy( 4, 4); write('PROBLEM:');
   gotoxy(31, 4); write('ACTION:');
   gotoxy(58,23); write('Today Is:'); textcolor(x)
end;


(* exposureform writes the field descriptions for the
   exposure record and one table which helps the user
   to fill the  fields of the  ranking method.  It is
   used by the database for updating exposures.       *)
procedure exposureform;
begin
  clearframe;
  gotoxy( 3, 6); write('Index:');
  gotoxy(16, 6); write('Description:');
  gotoxy( 5, 8); write('WEIGHTED:');
  gotoxy(15, 8); write('Damage:$');
  gotoxy(36, 8); write('Probability:');
  gotoxy( 5,10); write('P.E.R.T:');
  gotoxy(15,10); write('Smallest:$');
  gotoxy(36,10); write('Most Likely:$');
  gotoxy(60,10); write('Largest:$');
  gotoxy( 5,12); write('RANKS:');
  gotoxy(15,12); write('Rank P:');
  gotoxy(36,12); write('Rank Q:');
  textcolor(7);  gotoxy( 5,13);
  write('Rank P Damage caused by error');
  gotoxy(45,13);
  write('Rank Q Damage caused by failure');
  gotoxy( 8,14);
  write('0   virtually impossible');
  gotoxy(48,14);
  write('0   negligible');
  gotoxy( 8,15);
  write('1   might happen once in 400 years');
  gotoxy(48,15);
  write('1   about         $10');
  gotoxy( 8,16);
  write('2   might happen once in  40 years');
  gotoxy(48,16);
  write('2   about        $100');
  gotoxy( 8,17);
  write('3   might happen once in   4 years');
  gotoxy(48,17);
  write('3   about      $1,000');
  gotoxy( 8,18);
  write('4   might happen once in 100  days');
  gotoxy(48,18);
  write('4   about     $10,000');
```

```
  gotoxy( 8,19);
  write('5   might happen once in  10  days');
  gotoxy(48,19);
  write('5   about    $100,000');
  gotoxy( 8,20);
  write('6   might happen once in   1   day');
  gotoxy(48,20);
  write('6   about  $1,000,000');
  gotoxy( 8,21);
  write('7   might happen ten  times a  day');
  gotoxy(48,21);
  write('7   over   $1,000,000');
  textcolor(x)
end;


(* exposurefields gives in inversed video the fields
   to be  filled for the exposure record              *)
procedure exposurefields;
begin
  gotoxy(9,6); write('  ');
  textbackground(14);
  textcolor(0);
  gotoxy(28, 6); write(conststr(' ',50));
  gotoxy(23, 8); write(conststr(' ',8));
  gotoxy(48, 8); write('0.   ');
  gotoxy(25,10); write(conststr(' ',8));
  gotoxy(49,10); write(conststr(' ',8));
  gotoxy(69,10); write(conststr(' ',8));
  gotoxy(22,12); write(' .   ');
  gotoxy(43,12); write(' .   ');
  textbackground(z);
  textcolor(x)
end;


(* controlform  writes the field  descriptions for the
   control record. The number of its fields depends on
   the number of exposures. It is used by the database
   for updating controls.                             *)
procedure controlform( var expno : integer);
var
  i : integer;
begin
  clearframe;
  gotoxy( 3, 7); write('Index:');
  gotoxy(16, 7); write('Description:');
  gotoxy(22, 8); write('Cost:$');
  for  i := 1 to expno  do
  begin
    if  i <= 12  then
      gotoxy(3,9+i)
```

```
    else
      gotoxy(43,i-3);
    write('Effectiveness on Exposure ',i:2,':')
  end; {of for}
end;


(* controlfields gives the fields to be filled
   for the  control record in inversed video                    *)
procedure controlfields(expno : integer);
var
  i : integer;
begin
  gotoxy(9,7); write('  ');
  textbackground(14);
  textcolor(0);
  gotoxy(28, 7); write(conststr(' ',50));
  gotoxy(28, 8); write(conststr(' ',8));
  for  i := 1 to expno  do
  begin
    if  i <= 12  then
      gotoxy(33,9+i)
    else
      gotoxy(73,i-3);
    write('0.   ');
  end; {of for}
  textbackground(z);
  textcolor(x)
end;
```

```
(*************************************************************)
(*                                                           *)
(*                      DATABASE.DSS                         *)
(*                                                           *)
(*   This is the database of the system and performs all     *)
(*   the functions contained in the dbasemenu.               *)
(*************************************************************)


overlay procedure database;
label
  cancel;
var
  ans   : char;
  next  : integer;

(* makeproblem  creates the  control and  exposure files
   for each new problem and puts the problem description
   in the problem area of the frame.                      *)
overlay procedure makeproblem(s : str8);
begin
  clearframe;
  problemfield(s);
  action('NEW PROBLEM');
  message('CREATING    EXPOSURE    AND    CONTROL   FILES');
  delay(2000);
  makefile(file1,dr+s+'.dxp',sizeof(expsr));
  makeindex(index1,dr+s+'.ixp',sizeof(expsr.index),0);
  closefile(file1);
  closeindex(index1);
  makefile(file1,dr+s+'.dcl',sizeof(ctrl));
  makeindex(index1,dr+s+'.icl',sizeof(ctrl.index),0);
  closefile(file1);
  closeindex(index1);
end;

(* deleteproblem deletes all  the files  referred to the
   current problem,  removes  its  description  from the
   problem area and removes also the record  referred to
   that from the directory of the system.                 *)
overlay procedure deleteproblem(s : str8);
var
  i : integer;

begin
  clearmessage;
  clearselect;
  action('DELETING PROBLEM');
  assign(fl,dr+s+'.dxp');
  erase(fl);
```

```
assign(fl,dr+s+'.ixp');
erase(fl);
assign(fl,dr+s+'.dcl');
erase(fl);
assign(fl,dr+s+'.icl');
erase(fl);
openfile(file1,dr+s+'.wdt',sizeof(st));
if  ok  then
begin
  closefile(file1);
  assign(fl,dr+s+'.wdt');
  erase(fl);
  assign(fl,dr+s+'.wic');
  erase(fl)
end;
openfile(file1,dr+s+'.pdt',sizeof(st));
if  ok  then
begin
  closefile(file1);
  assign(fl,dr+s+'.pdt');
  erase(fl);
  assign(fl,dr+s+'.pic');
  erase(fl)
end;
openfile(file1,dr+s+'.rdt',sizeof(st));
if  ok  then
begin
  closefile(file1);
  assign(fl,dr+s+'.rdt');
  erase(fl);
  assign(fl,dr+s+'.ric');
  erase(fl)
end;

(* delete the directory of the current drive if it
   does not contain onother problem                    *)
openfile(file2,dr+'problem.dta',sizeof(problem));
if  usedrecs(file2) > 1  then
begin
  initindex;
  openindex(index2,dr+'problem.idx',sizeof(cproblem),0);
  deletekey(index2,i,s);
  deleterec(file2,i);
  closefile(file2);
  closeindex(index2)
end
else
begin
  closefile(file2);
  assign(fl,dr+'problem.dta');
  erase(fl);
  assign(fl,dr+'problem.idx');
```

```
    erase(f1)
  end;
  clearproblem
end;


(* updatecontrol adds, deletes, edits and scrolls
   the file of the controls data.                    *)
overlay procedure  updatecontrol( cproblem  : str8;
                                  expno : integer );
label
  cancel;
var
  rn,i,t  : integer;
  idx     : string[2];
  ans     : char;


(*  writecontrol writes the content of a control record
    on the  input/output control form.                  *)
procedure writecontrol(ctrl  :  control;  expno  : integer);
var
  i : integer;
begin
  controlfields(expno);
  with  ctrl  do
  begin
    gotoxy(9, 7); write(index);
    textbackground(14);
    textcolor(0);
    gotoxy(28, 7); write(description);
    gotoxy(28, 8); write(cost);
    for  i := 1 to expno  do
    begin
      if  i <= 12  then
        gotoxy(33,9+i)
      else
        gotoxy(73,i-3);
      write(effect[i])
    end
  end;
  textbackground(z);
  textcolor(x)
end;
```

```
(* IOcontrol reads input data from the screen. It is
   used for adding and editing controls.              *)
procedure IOcontrol(var ctrl  : control;
                        ch    : char;
                        expno : integer );
var
  tc       : char;
  i,j,n,ti : integer;
  t1       : string[2];
  t2       : string[3];
  s        : chset;
begin
  fillchar(t1,sizeof(t1),0);
  t1 := '0.';
  s  := [#48..#57];
  n  := 2 + expno;
  tc := ' ';
  with  ctrl  do
  while  tc <> 'Y'  do
  begin
    i := 1;
    case  ch  of
      'A' : begin
              fillchar(ctrl,sizeof(ctrl),0);
              controlfields(expno);
              index := inttostr(usedrecs(file1)+1);
              if  length(index) = 1  then
              insert('0',index,1);
              gotoxy(9,7); write(index)
            end;
      'E' : begin
              writecontrol(ctrl,expno);
              if  next = 2  then
                i := expno+2
            end
    end; {of case}
    repeat
       case  i  of
        1       : inputstr(description,50,28,7,
                                [#32..#126],tc);
        2       : inputstr(cost,8,28,8,s,tc);
        3..14   : begin
                    fillchar(t2,sizeof(t2),0);
                    t2 := copy(effect[i-2],3,5);
                    inputstr(t2,3,35,7+i,s,tc)
                  end;
        15..26 : begin
                     fillchar(t2,sizeof(t2),0);
                   t2 := copy(effect[i-2],3,5);
                   inputstr(t2,3,75,i-5,s,tc)
                 end
      end; {of case}
```

```
      if  i > 2  then
      begin
        if    (length(t2)  <  3)  and  (length(t2) > 0) then
        for j := length(t2)+1 to 3 do
          insert('0',t2,j);
        effect[i-2] := t1 + t2
      end;
      ti := i;
      funckey(tc,i);
      if  (ti =  i)  and  (chr(ord(tc)-100)  <>  'H')   then
        i := i + 1
    until  i > n;
    select('IS RECORD CORRECT(Y/N)? :',
                  ['Y','y','N','n'],tc);
    clearselect
  end (of while)
end;


(* deletecontrol deletes the current control record,
   if there are more than  two controls in the file,
   on the screen and  adjusts the  index  of all the
   successor records in the file.                    *)
procedure  deletecontrol( s : str8; idx  : str2 );
var
  i,t,rn  : integer;
  tc      : char;
  tdx     : string[2];

begin
  clearframe;
  if  usedrecs(file1) = 2  then
  begin
    message('SYSTEM   REQUIRES   2   CONTROLS   AT  LEAST');
    wait
  end
  else
  begin
    message('****** DELETING      CONTROL  '+idx+' ******');
    deletekey(index1,rn,idx);
    deleterec(file1,rn);
    t := strtoint(idx);

    if  t <= usedrecs(file1)  then
    with  ctrl  do
    begin
      fillchar(tdx,sizeof(tdx),0);
      tdx := inttostr(t+1);
      if  length(tdx) = 1  then
        insert('0',tdx,1);
      findkey(index1,rn,tdx);
```

```
      repeat
        getrec(file1,rn,ctrl);
        t := strtoint(index) - 1;
        index := inttostr(t);
        if  length(index) = 1  then
          insert('0',index,1);
        putrec(file1,rn,ctrl);
        deletekey(index1,rn,tdx);
        addkey(index1,rn,index);
        nextkey(index1,rn,tdx)
      until  not ok;
      closeindex(index1);
      initindex;
      openindex(index1,cproblem+'.icl',sizeof(index),0)
    end {of if/with}
  end {of else}
end;


begin   {of updatecontrol}
  controlform(expno);
  initindex;
  openfile(file1,dr+cproblem+'.dcl',sizeof(ctrl));
  openindex(index1,dr+cproblem+'.icl',sizeof(ctrl.index),0);
  fillchar(idx,sizeof(idx),0);
  fillchar(ctrl,sizeof(ctrl),0);
  clearkey(index1);
  if  next <> 1  then
  begin
    nextkey(index1,rn,idx);
    if  ok  then
    begin
      getrec(file1,rn,ctrl);
      writecontrol(ctrl,expno)
    end
    else  goto cancel
  end;
  ans := ' ';

  with  ctrl  do
  while  ans <> 'Q' do
  begin
    action('UPDATE  CONTROLS');
    if  next = 2  then
      ans  := 'E'
    else
    begin
      flag := false;
      if  usedrecs(file1) >= 2  then
      begin
        select('A)dd, D)elete, E)dit, N)ext, P)revious ',
               'or Q)uit:',['A','D','E','N','P','Q'],ans);
```

```
    clearselect
  end
  else
    ans := 'A'
end;

case  ans  of
  'A' : begin
          action('ADD CONTROL ');
          if  usedrecs(file1) = maxctrl  then
          begin
            clearframe;
            message('THE  SYSTEM  CANNOT  HOLD ANOTHER',
                                          ' CONTROL');
            wait;
            goto cancel
          end;
          IOcontrol(ctrl,'A',expno);
          addrec(file1,rn,ctrl);
          addkey(index1,rn,index);
          idx := index;
          findkey(index1,rn,idx)
        end;
  'D' : begin
          action('DELETE CONTROL ');
          deletecontrol(cproblem,idx);
          controlform(expno);
          i := strtoint(idx);
          clearkey(index1);
          if  i <= usedrecs(file1)  then
            findkey(index1,rn,idx)
          else
          repeat
            nextkey(index1,rn,idx)
          until ok
        end;
  'E' : begin
          action('EDIT CONTROL');
          getrec(file1,rn,ctrl);
          IOcontrol(ctrl,'E',expno);
          putrec(file1,rn,ctrl)
        end;
  'N' : repeat
          nextkey(index1,rn,idx)
        until  ok;
  'P' : repeat
          prevkey(index1,rn,idx)
       until  ok
end; {of case}
```

```
    if  ans in ['D','N','P']  then
    begin
      getrec(file1,rn,ctrl);
      writecontrol(ctrl,expno)
    end;

    if  next = 2  then
    begin
      nextkey(index1,rn,idx);
      if  not ok  then
      begin
        next := 0;
        ans  := 'Q'
      end
    end

  end; {of with/while}
  cancel:  closefile(file1);
           closeindex(index1);

end;


(* updatexposure adds, deletes, edits and scrolls
   the file of the exposures data.                    *)
overlay  procedure    updatexposure(        cproblem : str8:

                                    var  expno   :  integer  );
label
  cancel;
var
  rn,i,t   : integer;
  idx      : string[2];
  ans      : char;


(*  writexposure writes the content of an exposure record
    on the  input/output exposure form.                  *)
procedure writexposure(expsr : exposure);
begin
  exposurefields;
  with  expsr  do
  begin
    gotoxy(9, 6); write(index);
    textbackground(14);
    textcolor(0);
    gotoxy(28, 6); write(description);
    gotoxy(23, 8); write(damage);
    gotoxy(48, 8); write(probability);
    gotoxy(25,10); write(smallest);
    gotoxy(49,10); write(mostlikely);
    gotoxy(69,10); write(largest);
```

```
    gotoxy(22,12); write(rankP);
    gotoxy(43,12); write(rankQ)
  end; {of with}
  textbackground(z);
  textcolor(x)
end;


(* IOexposure reads input data from the screen. It is
   used for adding and editing exposures.              *)
procedure  IOexposure(var  expsr  :  exposure;  ch  : char);
var
  tc      : char;
  t1      : string[1];
  t2      : string[3];
  i,j,ti  : integer;

begin
  i  := 1;
  tc := #0;
  with  expsr  do
  while  tc <> 'Y'  do
  begin
    i := 1;
    case  ch  of
      'A' : begin
              fillchar(expsr,sizeof(expsr),0);
              exposurefields;
              index := inttostr(usedrecs(file1)+1);
              if  length(index) = 1  then
              insert('0',index,1);
              gotoxy(9,6); write(index)
            end;
      'E' : writexposure(expsr);
    end; {of case}

    repeat

      case  i  of
        1   :  inputstr(description,50,28,6,[#32..#126],tc);

        2 : inputstr(damage,8,23,8,[#48..#57],tc);
        3 : begin
              fillchar(t2,sizeof(t2),0);
              t2 := copy(probability,3,3);
              inputstr(t2,3,50,8,[#48..#57],tc);
              if  (length(t2) < 3) and (length(t2) > 0) then

              for j := length(t2)+1 to 3 do
                insert('0',t2,j);
              probability := '0.' + t2;
            end;
```

```
        4     :    inputstr(smallest,8,25,10,[#48..#57],tc);

        5     :   inputstr(mostlikely,8,49,10,[#48..#57],tc);

        6 : inputstr(largest,8,69,10,[#48..#57],tc);
        7 : begin
              fillchar(t1,sizeof(t1),0);
              fillchar(t2,sizeof(t2),0);
              t1 := copy(rankP,1,1);
              t2 := copy(rankP,3,3);
              inputstr(t1,1,22,12,[#48..#57],tc);
              inputstr(t2,3,24,12,[#48..#57],tc);
              if  (length(t2) < 3) and (length(t2) > 0) then
              for j := length(t2)+1 to 3 do
                insert('0',t2,j);
              rankP := t1 + '.' + t2
            end;
        8 : begin
              fillchar(t1,sizeof(t1),0);
              fillchar(t2,sizeof(t2),0);
              t1 := copy(rankQ,1,1);
              t2 := copy(rankQ,3,3);
              inputstr(t1,1,43,12,[#48..#57],tc);
              inputstr(t2,3,45,12,[#48..#57],tc);
              if  (length(t2) < 3) and (length(t2) > 0) then
              for j := length(t2)+1 to 3 do
                insert('0',t2,j);
              rankQ := t1 + '.' + t2
            end
      end; {of case}

      ti := i;
      funckey(tc,i);
      if    (ti  =  i)  and  (chr(ord(tc)-100) <> 'H')  then
        i := i + 1
    until  i > 8;

    select('IS RECORD CORRECT(Y/N)? :',
                      ['Y','y','N','n'],tc);
    clearselect
  end; {of while}
end;
```

```
(* deletexposure  deletes the  current  control record
   on the screen, if there are more than two exposures
   in the exposure file and adjusts the  index  of all
   the  successor records.  Then it  opens the control
   file  and removes  from all the control records the
   reference to the deleted exposure.                  *)
procedure   deletexposure( cproblem  : str8;  idx  : str2 );
var
  i,rn,usdr,
  t,recno    : integer;
  tc         : char;
  tdx        : string[2];

begin
  clearframe;
  usdr := usedrecs(file1);
  if  usdr = 2  then
  begin
    message('SYSTEM  REQUIRES  2  EXPOSURES    AT   LEAST');
    wait
  end
  else
  begin
    message('******  DELETING    EXPOSURE  '+idx+' ******');
    deletekey(index1,rn,idx);
    deleterec(file1,rn);
    recno := strtoint(idx);
    usdr := usedrecs(file1);
    if  recno <= usdr  then
    with  expsr  do
    begin
      fillchar(tdx,sizeof(tdx),0);
      tdx := inttostr(recno+1);
      if  length(tdx) = 1  then
        insert('0',tdx,1);
      findkey(index1,rn,tdx);
      repeat
        getrec(file1,rn,expsr);
        t := strtoint(index) - 1;
        index := inttostr(t);
        if  length(index) = 1  then
          insert('0',index,1);
        putrec(file1,rn,expsr);
        deletekey(index1,rn,tdx);
        addkey(index1,rn,index);
        nextkey(index1,rn,tdx)
      until  not ok
    end; {of if/with}
    closefile(file1);
    closeindex(index1);
```

```
    with  ctrl  do
    begin
      initindex;
      openfile(file1,dr+cproblem+'.dcl',sizeof(ctrl));
      openindex(index1,dr+cproblem+'.icl',sizeof(index),0);
      clearkey(index1);
      repeat
        nextkey(index1,rn,tdx);
        if  ok   then
        begin
          getrec(file1,rn,ctrl);
          for  i := recno to usdr  do
          begin
            effect[i] := effect[i+1];
          end; {of for}
          fillchar(effect[i+1],6,0);
          putrec(file1,rn,ctrl);
        end
      until  not ok;
      closefile(file1);
      closeindex(index1)
    end; {of with}
    initindex;
    openfile(file1,dr+cproblem+'.dxp',sizeof(expsr));
    openindex(index1,dr+cproblem+'.ixp',
                      sizeof(expsr.index),0);
  end {of else}
end;


begin   {of updatexposure}
  exposureform;
  initindex;
  openfile(file1,dr+cproblem+'.dxp',sizeof(expsr));
  expno := usedrecs(file1);
  openindex(index1,dr+cproblem+'.ixp',
                    sizeof(expsr.index),0);
  fillchar(idx,sizeof(idx),0);
  fillchar(expsr,sizeof(expsr),0);
  clearkey(index1);
  if  next <> 1  then
  begin
    nextkey(index1,rn,idx);
    if  ok  then
    begin
      getrec(file1,rn,expsr);
      writexposure(expsr)
    end
    else  goto cancel
  end;
  ans := ' ';
```

```
with  expsr  do
while  ans <> 'Q'  do
begin
  action('UPDATE EXPOSURES');
  if  usedrecs(file1) >= 2  then
  begin
    select('A)dd, D)elete, E)dit, N)ext, P)revious ',
           'or Q)uit:',['A','D','E','N','P','Q'],ans);
    clearselect
  end
  else
    ans := 'A';

  case  ans  of
    'A' : begin
            action('ADD EXPOSURE');
            if  usedrecs(file1) = maxexp  then
            begin
              clearframe;
              message('THE  SYSTEM  CANNOT HOLD ANOTHER ',
                                                 'EXPOSURE');
              wait;
              goto cancel
            end;
            IOexposure(expsr,'A');
            addrec(file1,rn,expsr);
            addkey(index1,rn,index);
            idx := index;
            findkey(index1,rn,idx)
          end;
    'D' : begin
            action('DELETE EXPOSURE');
            deletexposure(cproblem,idx);
            exposureform;
            i := strtoint(idx);
            clearkey(index1);
            if  i <= usedrecs(file1)  then
              findkey(index1,rn,idx)
            else
            repeat
              nextkey(index1,rn,idx)
            until  ok
          end;
    'E' : begin
            action('EDIT EXPOSURE');
            getrec(file1,rn,expsr);
            IOexposure(expsr,'E');
            putrec(file1,rn,expsr)
          end;
    'N' : repeat
            nextkey(index1,rn,idx)
          until  ok;
```

```
      'P' : repeat
              prevkey(index1,rn,idx)
            until  ok
    end; {of case}

    if  (ans in ['D','N','P'])  then
    begin
      getrec(file1,rn,expsr);
      writexposure(expsr)
    end {of if}

  end; {of with/while}
  t := usedrecs(file1);
  if  expno < t  then
  begin
    expno := t;
    next := 2
  end
  else
    next := 0;
  expno := t;
  cancel: closefile(file1);
          closeindex(index1);
end;


(* get directory asks the user to define the drive he/she
   wants to use, writes  directory  in the work area  and
   asks for a problem description.                        *)
overlay procedure getdirectory;
label
  cancel;
var
  i, j, number : integer;
  tby          : real;
  ch           : char;
  idx          : string[2];

begin
  clearproblem;
  clearframe;
  message('DEFINE  THE  DRIVE  YOU  WANT TO USE FOR FILES');
  gotoxy(16,16);
  write('IT IS BETTER THE DSS TO BE ON  A DIFFERENT DRIVE');
  gotoxy(16,18);
  write('DO NOT USE THE LETTER C IF THERE IS NO HARD DISK');
  select('DRIVE  A,B,C,D,E  or F:',[#65..#70,#97..#102],ch);
  fillchar(dr,sizeof(dr),0);
  dr := ch + ':';
  clearframe;
  action('DIRECTORY');
```

```
openfile(file1,dr+'problem.dta',sizeof(problem));
if  ok  then
begin
  initindex;
  openindex(index1,dr+'problem.idx',
            sizeof(problem.problemname),0);
  clearkey(index1);
  i := 10;
  j := 1;
  gotoxy(12,7);
  write('CHOOSE ONE OF THE FOLLOWING OR CREATE YOUR ',
                                          'OWN PROBLEM');
  textbackground(14);
  textcolor(0);
  gotoxy(15,9);
  write('PROBLEM:');
  gotoxy(27,9);
  write('CREATED BY:');
  gotoxy(56,9);
  write('DATE:');
  textbackground(z);
  textcolor(x);
  nextkey(index1,number,problem.problemname);

  repeat
    getrec(file1,number,problem);
    gotoxy(15,i); write(problem.problemname);
    gotoxy(27,i); write(problem.creator);
    gotoxy(56,i); write(problem.date);
    i := i + 1;
    j := j + 1;
    if  (i > 20) and (usedrecs(file1) > j)  then
    begin
      i := 10;
      wait;
      cleartext
    end;
    nextkey(index1,number,problem.problemname)
  until  not ok;
  gotoxy(21,21);
  textbackground(3);
  textcolor(0);
  write('Number of Problems in the Directory: ',
                                  usedrecs(file1):2);
  textbackground(z);
  textcolor(x);
  closefile(file1);
  closeindex(index1)
end
else
begin
  spaceavailable(tby);
```

```
  if  tby < 30000.0  then
  begin
    message('THERE  IS  NOT  ENOUGH  SPACE ON DRIVE '+dr);
    wait;
    next := 3;
    goto cancel
  end;
  message('****** NEW  DIRECTORY ******');
  makefile(file1,dr+'problem.dta',sizeof(problem));
  makeindex(index1,dr+'problem.idx',
                    sizeof(problem.problemname),0);
  closefile(file1);
  closeindex(index1);
end;
initindex;
openfile(file1,dr+'problem.dta',sizeof(problem));
openindex(index1,dr+'problem.idx',
                  sizeof(problem.problemname),0);
action('GIVE PROBLEM NAME');
gotoxy(4,23);
write('ENTER THE NAME OF THE PROBLEM:');
fillchar(cproblem,sizeof(cproblem),0);
fillchar(problem,sizeof(problem),0);
inputstr(cproblem,8,35,23,[#48..#126],tc);
cproblem := upcasestr(cproblem);
adjuststr(cproblem);
findkey(index1,number,cproblem);
if  not ok  then
begin
  spaceavailable(tby);
  if  tby < 10000.0  then
  begin
    closefile(file1);
    closeindex(index1);
    clearframe;
    message('THERE IS NOT  ENOUGH  SPACE  ON  DRIVE '+dr);
    wait;
    next := 3;
    goto cancel
  end;
  clearselect;
  gotoxy(4,23); write('ENTER YOUR NAME:');
  inputstr(problem.creator,25,21,23,[#32..#126],tc);
  problem.problemname := cproblem;
  problem.creator := upcasestr(problem.creator);
  getdate(problem.date);
  number := 0;
  addrec(file1,number,problem);
  addkey(index1,number,problem.problemname);
  closefile(file1);
  closeindex(index1);
  expno := 0;
```

```
    next := 1;
  end
  else
  (* get key information about the latest model execution *)
  with  problem  do
  begin
    getrec(file1,number,problem);
    wcombindex := wcomb;
    pcombindex := pcomb;
    rcombindex := rcomb;
    wtotalcost := strtoreal(wtotcost);
    ptotalcost := strtoreal(ptotcost);
    rtotalcost := strtoreal(rtotcost);
    closefile(file1);
    closeindex(index1);
    openfile(file1,dr+cproblem+'.dxp',sizeof(expsr));
    expno := usedrecs(file1);
    closefile(file1);
    problemfield(cproblem)
  end;
  cancel:
end;


BEGIN  (OF DATABASE)
  ans := ' ';
  if  flag  then
  begin
    getdirectory;
    if  next = 1  then
    begin
      makeproblem(cproblem);
      updatexposure(cproblem,expno);
      next := 1;
      updatecontrol(cproblem,expno)
    end;
    if  next <> 3  then
      flag := false;
    next := 0;
    goto cancel
  end;

  while  ans <> '6'  do
  begin
    dbasemenu;
    select('SELECT 1,2,3,4,5 or 6 :',['1'..'6'],ans);
```

```
    case  ans  of
      '1' : help('D');
      '2' :begin
             getdirectory;
             if  next = 1  then
             begin
               makeproblem(cproblem);
               updatexposure(cproblem,expno);
               next := 1;
               updatecontrol(cproblem,expno);
               next := 0
             end
           end;
      '3' : begin
              clearframe;
              message('DO YOU WISH TO DELETE THE PROBLEM?');
              select('TYPE [!]  TO DELETE  OR ANY  KEY TO ',
                     'CANCEL',[#1..#126],ans);
              if  ans = '!' then
              begin
                deleteproblem(cproblem);
                flag := true;
                next := 0;
                goto cancel
              end;
              ans := ' '
            end;
      '4' : begin
              updatexposure(cproblem,expno);
              if  next = 2  then
              begin
                updatecontrol(cproblem,expno);
                next := 0
              end
            end;
      '5' : updatecontrol(cproblem,expno)
    end {of case}
  end; {of while}
  cancel:
END;
```

```
(*****************************************************************)
(*                                                               *)
(*                          MODEL.DSS                            *)
(*                                                               *)
(*    This is the model execution part of the system. The        *)
(*    user can select one or more statistical methods for        *)
(*    the model run.                                             *)
(*                                                               *)
(*****************************************************************)


overlay procedure  model;
label
  cancel,cont;
var
  ans,tans,ch   : char;
  flag1,flag2   : boolean;
  method        : string[2];
  expdam        : array[1..maxexp]  of real;
  i,rn,ctrlno   : integer;
  idx           : string[2];
  maximum       : string[10];
  benefit,
  u,y,t         : real;


(* weightedprobability computes the expected cost for
   each exposure of the exposure file.                    *)
overlay procedure  weightedprobability;
begin
  initindex;
  openfile(file1,dr+cproblem+'.dxp',sizeof(expsr));
  openindex(index1,dr+cproblem+'.ixp',
                          sizeof(expsr.index),0);
  clearkey(index1);
  fillchar(expdam,sizeof(expdam),0);
  fillchar(expsr,sizeof(expsr),0);
  totaloss := 0;
  with  expsr  do
  repeat
    nextkey(index1,rn,idx);
    if  ok  then
    begin
      getrec(file1,rn,expsr);
      i := strtoint(index);
      expdam[i] := strtoreal(damage) *
                   strtoreal(probability);
      totaloss := totaloss + expdam[i];
    end
  until  not ok;
```

```
  closefile(file1);
  closeindex(index1)
end;


(* pertmethod computes the expected cost for each exposure
   of the exposure file, using the P.E.R.T. method        *)
overlay procedure  pertmethod;
begin
  fillchar(expsr,sizeof(expsr),0);
  initindex;
  openfile(file1,dr+cproblem+'.dxp',sizeof(expsr));
  openindex(index1,dr+cproblem+'.ixp',
                              sizeof(expsr.index),0);
  fillchar(expdam,sizeof(expdam),0);
  clearkey(index1);
  totaloss := 0;
  with  expsr  do
  repeat
    nextkey(index1,rn,idx);
    if  ok  then
    begin
      getrec(file1,rn,expsr);
      i := strtoint(index);
      expdam[i] := (strtoreal(smallest) +
                    4 * strtoreal(mostlikely)
                    + strtoreal(largest)) / 6;
      totaloss := totaloss + expdam[i]
    end
  until  not ok;
  closefile(file1);
  closeindex(index1)
end;


(* rankingmethod computes the expected cost for each
   exposure of the exposure file,  using the Ranking
   method.                                            *)
overlay procedure  rankingmethod;
begin
  initindex;
  openfile(file1,dr+cproblem+'.dxp',sizeof(expsr));
  openindex(index1,dr+cproblem+'.ixp',
                              sizeof(expsr.index),0);
  clearkey(index1);
  fillchar(expdam,sizeof(expdam),0);
  fillchar(expsr,sizeof(expsr),0);
  totaloss := 0;
  y := ln(10);
```

```
  with  expsr  do
  repeat
    nextkey(index1,rn,idx);
    if  ok  then
    begin
      getrec(file1,rn,expsr);
      u :=  y *  (strtoreal(rankP) +  strtoreal(rankQ) - 3);
      i := strtoint(index);
      expdam[i] := exp(u) / 4.0;
      totaloss := totaloss + expdam[i]
    end
  until  not ok;
  closefile(file1);
  closeindex(index1)
end;


(* effectivecontrol computes the effectiveness for each
   control activity in the control file. If the control
   is an effective one then it is loaded  in memory for
   subsequent computation.                                  *)
 overlay procedure  effectivecontrol;
begin
  initindex;
  openfile(file1,dr+cproblem+'.dcl',sizeof(ctrl));
  openindex(index1,dr+cproblem+'.icl',sizeof(ctrl.index),0);
  clearkey(index1);
  fillchar(ctrlmatrix,sizeof(ctrlmatrix),0);
  fillchar(ctrl,sizeof(ctrl),0);
  fillchar(comb,sizeof(comb),0);
  ctrlno    := 0;
  totalcost := 0;
  with  ctrl  do
  repeat
    nextkey(index1,rn,idx);
    if  ok  then
    begin
      benefit   := 0;
      getrec(file1,rn,ctrl);
      for  i := 1 to expno  do
        benefit := benefit + strtoreal(effect[i]) *
                                          expdam[i];

      if  benefit > strtoreal(cost)  then
      begin
        ctrlno := ctrlno + 1;
        ctrlmatrix[ctrlno] := ctrl;
        totalcost  := totalcost + strtoreal(cost);
        comb[ctrlno] := index
      end
      else
      begin
```

```
        message('CONTROL '+ copy(description,1,
                 length(description))+'  IS NOT EFFECTIVE');
        delay(2000);
        clearmessage
      end
    end
  until  not ok;
  closefile(file1);
  closeindex(index1);
end;


(* controlsets generate all the possible control sets
   and updates the problem record in the directory of
   the system. It has the ability  also to create and
   delete the set files.                                  *)
overlay procedure  controlsets(maxcost : real);
label
  cancel;
var
  i,j,k,l,maxcomb,p,rn : integer;
  cost,value,seff,tby,
  filebytes,indexbytes : real;
  combination          : array[1..maxctrl] of char;

(* binary converts a decimal number to its binary
   representation. Its purpose is to generate the
   combinations of the control activities.        *)
procedure  binary(k : integer);
var
  j    : integer;
begin
  for  j := 1 to ctrlno  do
  begin
    if  k mod 2 <> 0  then
      combination[j] := '1'
    else
      combination[j] := '0';
    k := k div 2
  end {of for}
end;

begin
  maxcomb := round(exp(ln(2) * ctrlno)) - 1;

  (* computed the size in bytes of the set file *)
  filebytes  := (sizeof(st) * 1.0) * maxcomb;
  indexbytes := (((sizeof(st.Ck) + 5)* (order+3) * 1.0) *
                                          maxcomb)/order;
   (* ask the available bytes of the specified drive *)
  spaceavailable(tby);
```

```
if  tby < (filebytes+indexbytes)  then
begin
  message('THERE  IS   NOT  ENOUGH  SPACE  ON  DRIVE'+dr);
  flag2 := true;
  goto cancel
end;
textbackground(3);
initindex;
makefile(file1,dr+cproblem+'.'+method+'dt',sizeof(st));
makeindex(index1,dr+cproblem+'.'+method+'ic',
                                        sizeof(st.Ck),1);
p := 0;
with  st  do
for  i := 1 to maxcomb  do
begin
  fillchar(st,sizeof(st),0);
  fillchar(combination,sizeof(combination),0);
  binary(i);
  cost := 0.0;
  for  j := 1 to ctrlno  do
  if  combination[j] = '1'  then
    cost := cost + strtoreal(ctrlmatrix[j].cost);
  if  maxcost >= cost  then
  begin
    value := 0.0;
    for  j := 1 to expno  do
    begin
      seff := 0.0;
      for  l := 1 to ctrlno  do
      if  combination[l] = '1'  then
        seff := seff + (1 - seff) *
                strtoreal(ctrlmatrix[l].effect[j]);
      value := value + seff * expdam[j]
    end; {of for j}

    (* Keep only the effective control sets *)
    if  value > cost  then
    begin
      l := 0;
      for  j := 1 to ctrlno  do
        if  combination[j] = '1'  then
        begin
          l := l + 1;
          setcomb[l] := ctrlmatrix[j].index
        end;
      p := p + 1;
      gotoxy(31,21);   write('Number   of   Sets  :',p:4);
      str(value:10:0,Vk);
      str((totaloss - value):10:0,Lk);
      str(cost:10:0,Ck);
      str((value - cost):10:0,Nk);
      str((totaloss - value + cost):10:0,TCk);
```

```
      str((value / cost):5:4,BCR);
      if  strtoreal(BCR) < 10.0  then
        insert(' ',BCR,1);
      addrec(file1,rn,st);
      addkey(index1,rn,Ck);
    end
  end
end; {of for i}
gotoxy(31,21); write(conststr(' ',20));
if  usedrecs(file1) = 0  then
begin
  closefile(file1);
  closeindex(index1);
  assign(fl,dr+cproblem+'.'+method+'dt');
  erase(fl);
  assign(fl,dr+cproblem+'.'+method+'ic');
  erase(fl)
end
else
begin
  closefile(file1);
  closeindex(index1);
  if  totalcost > maxcost  then
    totalcost := maxcost;
  case  method  of
    'w' : begin
            wcombindex    := comb;
            problem.wcomb := comb;
            wtotalcost    := totalcost;
            str(totalcost:10:0,problem.wtotcost);
            adjuststr(problem.wtotcost)
          end;
    'p' : begin
            pcombindex    := comb;
            problem.pcomb := comb;
            ptotalcost    := totalcost;
            str(totalcost:10:0,problem.ptotcost);
            adjuststr(problem.ptotcost)
          end;
    'r' : begin
            rcombindex    := comb;
            problem.rcomb := comb;
            rtotalcost    := totalcost;
            str(totalcost:10:0,problem.rtotcost);
            adjuststr(problem.rtotcost)
          end
  end; {of case}
  openfile(file1,dr+'problem.dta',sizeof(problem));
  openindex(index1,dr+'problem.idx',
                   sizeof(problem.problemname),0);
  findkey(index1,rn,cproblem);
  putrec(file1,rn,problem);
```

```
    closefile(file1);
    closeindex(index1)
  end;
  cancel: textbackground(z)
end;

BEGIN   {OF MODEL}
  ans  := ' ';
  flag1 := false;
  while  ans <> '6'  do
  begin
    if  not flag1  then
    begin
      modelmenu;
      select('SELECT  1,2,3,4,5   or  6  :',['1'..'6'],ans);
      clearframe;
      tans := ans
    end;
    if  ans = '5'  then
    begin
      flag1 := true;
      ans  := '0';
      tans := '2'
    end;
    ch := #0;
    case  tans  of
      '1' : help('0');
      '2' : begin
              method := 'w';
              action('MODEL / WEIGHTED METHOD');
            end;
      '3' : begin
              method := 'p';
              action('MODEL / P.E.R.T. METHOD');
            end;
      '4' : begin
              method := 'r';
              action('MODEL / RANKING METHOD');
              flag1 := false
            end;
      '6' : goto  cancel
    end; {of case}
    if  tans in ['2'..'5']  then
    begin
      openfile(file1,dr+cproblem+'.'+method+'dt',
                                          sizeof(st));
      if  ok  then
      begin
        closefile(file1);
        message('THERE IS ALREADY  FILE  FOR  THAT METHOD');
        select('SELECT D)elete, R)un or any key to cancel:',
                                            [#1..#126],ch);
```

```
  clearmessage;
  clearselect;
end;
if  ch  in ['D','R',#0]  then
begin
  if  ch in ['D','R']  then
  begin
    assign(fl,dr+cproblem+'.'+method+'dt');
    erase(fl);
    assign(fl,dr+cproblem+'.'+method+'ic');
    erase(fl)
  end;
  if  ch = 'D'  then
    goto cont;
  case  tans  of
    '2' : weightedprobability;
    '3' : pertmethod;
    '4' : rankingmethod
  end;
  effectivecontrol;
  if totalcost  < 1.0  then
  begin
    message('CANNOT COMPUTE  SETS WITHOUT EFFECTIVE ',
                                          'CONTROLS');
    wait;
    goto cont
  end;
  if  ctrlno = 1  then
  begin
    message('CANNOT  COMPUTE  SETS  WITH  ONLY  ONE ',
                                  'EFFECTIVE CONTROL');
    wait;
    goto cont
  end;
  if  totaloss > totalcost  then
    t := totalcost
  else
    t := totaloss;
  gotoxy(20, 8);
  write('Total Damage Due To Exposures  :',
                                      totaloss:10:0);
  gotoxy(20,10);
  write('Cost to Implement All Controls :',
                                     totalcost:10:0);
  gotoxy(13,14);
  write('Give The Maximum Amount You Want To ',
                                  'Spend On Controls');
  gotoxy(29,15);  write('or  press  Enter  for  ALL');
  gotoxy(30,17); write('MAXIMUM : $');
  str(t:10:0,maximum);
  adjuststr(maximum);
  inputstr(maximum,10,42,17,['0'..'9'],ch);
```

```
        clearframe;
        message('PLEASE  WAIT');
        controlsets(strtoreal(maximum));
        if  flag2  then
        begin
          flag2 := false;
          goto cont
        end;
        clearmessage
      end
    end;
    cont: if  flag1  then
          begin
            tans := chr(ord(tans)+1);
            clearmessage
          end
  end; {of while}
  cancel:
END;
```

```
(*************************************************************)
(*                                                           *)
(*                      SENSANAL.DSS                         *)
(*                                                           *)
(*  This is the sensitivity analysis part of the system.     *)
(*  It consists of procedures for control strategy           *)
(*  selection, graphics and report printouts.                *)
(*                                                           *)
(*************************************************************)


overlay procedure sensitivityanalysis;
var
  a,b                : plotarray;
  i,j,k,rn,ctrlno,
  maxNKrn,maxBCRrn   : integer;
  maxNk,maxBCR,
  low,high,key       : string[10];
  idx                : string[2];
  ans,method,tc      : char;


(* inputlimits prompts the user to give the desired cost
   range within which the set files will be searched.     *)
procedure inputlimits;
begin
  clearframe;
  case  method  of
    'W' : totalcost := wtotalcost;
    'P' : totalcost := ptotalcost;
    'R' : totalcost := rtotalcost
  end;
  clearkey(index2);
  nextkey(index2,rn,key);
  getrec(file2,rn,st);
  adjuststr(st.Nk);
  adjuststr(st.TCk);
  totaloss := strtoreal(st.Nk) + strtoreal(st.TCk);
  gotoxy(19,7);
  write('Total Damage  Due To   Exposures :',totaloss:10:0);
  gotoxy(19,9);
  write('Maximum Cost in the Set File    :',totalcost:10:0);
  gotoxy(12,15);
  write('Give the  Cost Range  over which the Search will be
                                                    done:');
  gotoxy(29,17); write('Low  Limit : $');
  gotoxy(29,19); write('High Limit : $');
  repeat
    fillchar(low,10,0);
    fillchar(high,10,0);
```

```
    textbackground(14);
    textcolor(0);
    gotoxy(44,17); write(conststr(' ',10));
    gotoxy(44,19); write(conststr(' ',10));
    inputstr(low,10,44,17,['0'..'9'],tc);
    inputstr(high,10,44,19,['0'..'9'],tc);
    if  strtoreal(high) <= strtoreal(low)  then
    begin
      message('CHECK YOUR ENTRY. "HIGH" MUST BE GREATER '
                                              'THAN  "LOW"');
      wait;
      clearmessage
    end
  until  strtoreal(high) > strtoreal(low);
  while  length(low) < 10  do
    insert(' ',low,1);
  while  length(high) < 10  do
    insert(' ',high,1);


(* use opens the files most commonly used in the
   sensitivity analysis process.                    *)
procedure  use(cproblem : str8; method : char);
begin
  initindex;
  openfile(file1,dr+cproblem+'.dcl',sizeof(ctrl));
  openindex(index1,dr+cproblem+'.icl',sizeof(ctrl.index),0);
  openfile(file2,dr+cproblem+'.'+method+'dt',sizeof(st));
  openindex(index2,dr+cproblem+'.'+method+'ic',
                                        sizeof(st.Ck),1);
end;


(* closefiles closes files opened with the use procedure *)
procedure  closefiles;
begin
  closefile(file1);
  closeindex(index1);
  closefile(file2);
  closeindex(index2);
end;

overlay procedure  controlstrategy;
label
  cancel;
var
  tloss : real;
  title : string[60];
begin
  ans := ' ';
  while  ans <> '4'  do
  begin
```

```
controlstrategymenu;
select('SELECT 1,2,3 or 4 : ',['1'..'4'],ans);
if  ans = '1'  then
  help('B')
else
while  ans <> '4'  do
begin
  clearframe;
  select('SELECT W)eighted, P).e.r.t, R)anking
                  or   Q)uit:',['W','P','Q','R'],method);
  clearselect;
  if  method = 'Q'  then goto cancel;

  openfile(file1,dr+cproblem+'.'+method+'dt',
                                      sizeof(st));
  if  not  ok  then
  begin
    clearframe;
    message('YOU MUST RUN THE MODEL FIRST');
    wait;
    goto cancel
  end;
  fillchar(maxNk,10,0);
  fillchar(maxBCR,10,0);
  case  method  of
    'W' : title := 'WEIGHTED METHOD: ';
    'P' : title := 'P.E.R.T. METHOD: ';
    'R' : title := 'RANKING  METHOD: ';
  end; {of case}
  closefile(file1);
  use(cproblem,method);
  inputlimits;
  key := low;
  searchkey(index2,rn,key);
  if  ok and (key <= high)  then
  with  st  do
  begin
    repeat
      getrec(file2,rn,st);
      adjuststr(Nk);
      adjuststr(BCR);
      if  strtoreal(Nk) > strtoreal(maxNk)  then
      begin
        maxNk   := Nk;
        maxNkrn := rn
      end;
```

```
  if    strtoreal(BCR)  >  strtoreal(maxBCR)    then

  begin
    maxBCR   := BCR;
    maxBCRrn := rn
  end;
  nextkey(index2,rn,key)
until  not ok or (key > high);
if  ans = '2'  then
begin
  title := title + 'THE MOST EFFECTIVE SET';
  rn := maxNkrn
end;
if  ans = '3'  then
begin
  title :=  title +  'THE MOST  COST EFFECTIVE SET';
  rn := maxBCRrn
end;
clearframe;
fillchar(st,sizeof(st),0);
gotoxy(10,6); write(title);
getrec(file2,rn,st);
j := 7;
for  i := 1 to maxctrl  do
if  setcomb[i] <> ''  then
begin
  j := j + 1;
  findkey(index1,rn,setcomb[i]);
  getrec(file1,rn,ctrl);
  gotoxy(10,j); write('CONTROL ',idx,': ',
                          ctrl.description)
end;
if  j+8 > 21  then
begin
  wait;
  clearframe;
  j := 7
end;
gotoxy( 3,j+2);write('Value  of Control Set  :',Vk);
gotoxy(43,j+2);write('Cost  of Control  Set  :',Ck);
gotoxy(  3,j+3);write('Total Expected Benefit:',Nk);
gotoxy(43,j+3);write('Total Expected  Cost  :',TCk);
gotoxy(25,j+5);write('Benefit Cost  Ratio   :',BCR);
gotoxy(12,j+7);
write('Prior Expected Damage Due to Exposures:',
                                    totaloss:8:0);
```

```
        adjuststr(Vk);
        tloss := totaloss - strtoreal(Vk);
        gotoxy(12,j+8);
        write('Post  Expected Damage Due to Exposures:',
                                                  tloss:8:0);
        wait;
        closefiles
      end {of if/with}
      else
      begin
        message('THERE  IS  NO  ANY SET WITHIN THAT RANGE');
        wait
      end
    end; {of while}
    cancel:
  end {of while}
end;
```

```
overlay procedure  graphics;
label
  cancel;
var
  ans              : char;
  title            : string[16];
  currentaction    : string[40];
  Y1min,Y1max,
  Y2min,Y2max,
  temp1,temp2,
  prevCk,prevBCR,
  prevTCK          : real;
  bestset1,
  bestset2,i       : integer;
  flag1,flag2      :boolean;

(* computegraph computes the values of the plotarrays
   which will be  used by the  makegraph procedure to
   draw the graphs.                                    *)
overlay procedure  computegraph;
var
  ti : integer;
begin
  if  flag2  then
    ti := MaxPlotGlb
  else
    ti := 24;
  use(cproblem,method);
  inputlimits;
  message('PLEASE WAIT  FOR THE  PREPARATION OF THE GRAPH');
  i := 0;
  key := low;
  searchkey(index2,rn,key);
  Y1min := 9.99E+20;
  Y1max := 0.0;
  Y2min := 9.99E+20;
  Y2max := 0.0;
  if  ok and (key <= high)  then
  with  st  do
  begin
    repeat
      if  i < ti  then
      begin
        getrec(file2,rn,st);
        adjuststr(key);
        adjuststr(BCR);
        adjuststr(TCk);
        temp1 :=  strtoreal(BCR);
        temp2 := strtoreal(TCk);
```

```
        if  strtoreal(key) = a[i,1]  then
        begin
          if  temp1 > a[i,2]  then
            a[i,2] := temp1;
          if  temp2 < b[i,2]  then
            b[i,2] := temp2
        end
        else
        begin
          i := i + 1;
          a[i,1] := strtoreal(key);
          b[i,1] := strtoreal(key);
          a[i,2] := temp1;
          b[i,2] := temp2
        end;
        if  Y1max < temp1  then
        begin
          Y1max    := temp1;
          bestset1 := rn
        end;
        if  Y1min > temp1  then
          Y1min := temp1;
        if  Y2min > temp2  then
        begin
          Y2min := temp2;
          bestset2 := rn
        end;
        if Y2max < temp2  then
          Y2max := temp2
      end;
      nextkey(index2,rn,key)
    until  not ok or (i = ti) or (key > high);
    if  (i = ti) and (key < high) and ok then
    begin
      high := key;
      message('Cannot Graph  All Sets. Cost Range Has Been',
                                                ' Adjusted');
      wait
    end
  end {of if/with}
  else
  begin
    message('THERE  IS  NO  ANY  SET  WITHIN  THAT  RANGE');
    flag1 := true;
    wait;
    clearframe
  end
end;
```

```
overlay procedure  makegraph;
var
  j       : integer;
  step    : real;
  numtext : string[7];
begin
  if  i < 2  then
    message('CANNOT  MAKE  GRAPH  WITH  LESS  THAN  2 SETS')
  else
  begin
    initgraphic;
    setbreakoff;
    setmessageoff;
    setlinestyle(0);
    setforegroundcolor(0);
    (* draw the first graph (upper left side) *)
    definewindow(1,0,0,trunc(Xmaxglb/1.5),trunc(Ymaxglb/2));
    defineheader(1,'BENEFIT COST RATIO VS COST FOR '+title);
    if  flag2  then
    begin
      defineworld(1,a[1,1]/1.02,Y1min/1.1,a[i,1]*1.02,
                                                Y1max*1.1);
      selectwindow(1);
      selectworld(1);
      setheaderon;
      setbackground(0);
      drawborder;
      drawaxis(9,9,0,0,0,0,0,0,false);
      drawpolygon(a,1,i,4,1,0)
    end
    else
    begin
      defineworld(1,a[1,1],Y1min/1.1,a[i,1],Y1max*1.2);
      selectwindow(1);
      selectworld(1);
      setheaderon;
      setbackground(0);
      drawborder;
      drawhistogram(a,i,true,4);
      drawtextW(a[1,1],Y1max*0.07 + Y1min/1.1,1,
      'Costs below  are in Thousands of Dollars (rounded)');
      fillchar(numtext,sizeof(numtext),0);
      step := (a[i,1]-a[1,1]) / i;
      for  j := 1 to i  do
      begin
        str(round(a[j,1]/1000):7,numtext);
        adjuststr(numtext);
        drawtextW(a[1,1]+step*(j-1),Y1max*0.18+Y1min/1.1,1,
                        ' '+ copy(numtext,1,length(numtext)))
      end
    end;
```

```
(* draw the second graph at the lower left side
   of the screen.                                    *)
definewindow(2,trunc(Xmaxglb/3),trunc(Ymaxglb/2),
                                    Xmaxglb,Ymaxglb);
defineheader(2,'TOTAL EXPECTED COST VS COST OF CONTROL',
                                          ' / '+ title);
if  flag2  then
begin
  flag2 := false;
  defineworld(2,b[1,1]/1.02,Y2min/1.02,b[i,1]*1.02,
                                       Y2max*1.02);
  selectwindow(2);
  selectworld(2);
  setheaderon;
  setbackground(0);
  drawborder;
  drawaxis(9,9,0,0,0,0,0,0,false);
  drawpolygon(b,1,i,4,1,0)
end
else
begin
  defineworld(2,b[1,1],Y2min/1.1,b[i,1],Y2max*1.2);
  selectwindow(2);
  selectworld(2);
  setheaderon;
  setbackground(0);
  drawborder;
  drawhistogram(b,i,true,4);
  drawtextW(b[1,1],Y2max*0.07 + Y2min/1.1,1,
  'Costs below are in Thousands of  Dollars (rounded)');
  fillchar(numtext,sizeof(numtext),0);
  step := (b[i,1]-b[1,1]) / i;
  for  j := 1 to i  do
  begin
    str(round(b[j,1]/1000):7,numtext);
    adjuststr(numtext);
    drawtextW(b[1,1]+step*(j-1),Y2max*0.18+Y2min/1.1,1,
                  ' '+ copy(numtext,1,length(numtext)))
  end
end;
gotoxy(55,1); write('GRAPHS OVER THE RANGE:');
gotoxy(59,2); write('Low :'+ low);
gotoxy(59,3); write('High:'+ high);
gotoxy(55,4); write('Number of Sets :',i:3);
getrec(file2,bestset1,st);
adjuststr(st.BCR);
adjuststr(st.Ck);
gotoxy(60,8); write('<<== THE BEST SET ');
gotoxy(59,9); write('BCR          : ',st.BCR);
gotoxy(59,10); write('Cost of set : ',st.Ck);
```

```
    getrec(file2,bestset2,st);
    adjuststr(st.TCk);
    adjuststr(st.Ck);
    gotoxy(5,18); write('THE BEST SET ==>>');
    gotoxy(1,20); write('Expected cost: ',st.TCk);
    gotoxy(1,21); write('Cost  of  set: ',st.Ck);
    gotoxy(1,24); write('press any key ...');
    setforegroundcolor(2);
    read(kbd,ans);
    leavegraphic;
    textmode;
    textcolor(x);
    makeframe;
    problemfield(cproblem);
    action(currentaction);
    putdate
  end;
  closefiles;
end;

begin  { of graphics}
  ans  := ' ';
  flag1 := false;
  flag2 := false;

  while  ans <> '4'  do
  begin
    graphicsmenu;
    select('SELECT  1,2,3 or 4 : ',['1'..'4'],ans);
    if  ans = '1'  then
      help('G');

    while  (ans <> '1') and (ans <> '4')  do
    begin
      select('SELECT W)eighted, P).e.r.t, R)anking
              or Q)uit: ',['W','P','Q','R'],method);
      if  method = 'Q'  then  goto cancel;
      openfile(file1,dr+cproblem+'.'+method+'dt',
                                        sizeof(st));

      if  not  ok  then
      begin
        clearframe;
        message('YOU MUST RUN THE MODEL FIRST');
        wait;
        goto  cancel
      end;
```

```
      if  ok  then
      begin
        if  usedrecs(file1) < 2  then
        begin
          clearframe;
          message('CANNOT  MAKE  GRAPH  WITH  LESS  THAN  2
                                                SETS');
          closefile(file1);
          wait;
          closefile(file1);
          goto cancel
        end;
        closefile(file1);

        case  method  of
          'W' : title := 'WEIGHTED METHOD ';
          'P' : title := 'P.E.R.T. METHOD ';
          'R' : title := 'RANKING  METHOD ';
        end {of case}

      end; {of if ok}

      case  ans  of
        '2' : begin
                currentaction := 'GRAPHICS / CURVE';
                action(currentaction);
                flag2 := true;
                computegraph;
                if  not flag1  then
                  makegraph;
                flag1 := false;
                ans := '2'
              end;
        '3' : begin
                currentaction  :=  'GRAPHICS  /  HISTOGRAM';
                action(currentaction);
                computegraph;
                if  not flag1  then
                  makegraph;
                flag1 := false;
                ans := '3'
              end
      end {of case}

    end; {of while}
    cancel:
  end {of while}
end;
```

```
overlay procedure printfiles;
var
  ans : char;

overlay procedure  controleffectable;
var
  header : string[80];
  idx    : string[2];
  i      : integer;
begin
  action('PRINTER / CONTROL TABLE');
  fillchar(ce,sizeof(ce),0);
  fillchar(header,sizeof(header),0);
  writeln(lst,#12,'D E C I S I O N   S U P P O R T  ',
                                    'S Y S T E M':62);
  writeln(lst,conststr('-',47):63);
  writeln(lst);
  writeln(lst,'COST  EFFECTIVENESS  ANALYSIS':54);
  writeln(lst,'FOR':41);
  writeln(lst,'CONTROL  &  SECURITY  OF  COMPUTER  ',
                                       'SYSTEMS':62);
  writeln(lst);
  writeln(lst);
  writeln(lst,'CONTROL  ACTIVITIES   FOR  WORK  '+cproblem);

  openfile(file1,dr+cproblem+'.dcl',sizeof(ctrl));
  openindex(index1,dr+cproblem+'.icl',
                  sizeof(ctrl.index),0);
  ctrlno := usedrecs(file1);
  clearkey(index1);
  header := 'EXPOSURE |  ';
  i := 0;
  repeat
    nextkey(index1,rn,idx);
    if  ok  then
    begin
      i := i + 1;
      getrec(file1,rn,ctrl);
      writeln(lst,ctrl.index,'  ',ctrl.description);
      ce[i] := ctrl.effect;
      cc[i] := ctrl.cost;
      header := header + ctrl.index +'  |  '
    end
  until  not ok;
  closefile(file1);
  closeindex(index1);
  writeln(lst);
  writeln(lst);
  writeln(lst,'EXPOSURES FOR WORK '+cproblem);
```

```
  openfile(file1,dr+cproblem+'.dxp',sizeof(expsr));
  openindex(index1,dr+cproblem+'.ixp',
                              sizeof(expsr.index),0);
  clearkey(index1);
  repeat
    nextkey(index1,rn,idx);
    if  ok  then
    begin
      getrec(file1,rn,expsr);
      writeln(lst,expsr.index,'          ',expsr.description)
    end
  until  not ok;
  closefile(file1);
  closeindex(index1);
  writeln(lst);
  writeln(lst);
  writeln(lst);
  writeln(lst,conststr('=',80));
  writeln(lst,'EFFECTIVENESS OF CONTROL a(i)  ON EXPOSURE ',
                                              'e(i)':70);
  writeln(lst,header);
  writeln(lst,conststr('-',80));
  for  i := 1 to expno  do
  begin
    write(lst,i:6,'    !');
    for  j := 1 to ctrlno  do
      write(lst,ce[j,i]:6,' ');
    writeln(lst);
  end;
  writeln(lst);
  write(lst,'COST a(i)!');
  for  i := 1 to ctrlno do
    write(lst,strtoreal(cc[i]):6:0,' ');
  writeln(lst);
  writeln(lst,conststr('=',80))
end;


overlay procedure exposuretable;
begin
  action('PRINTER / EXPOSURE TABLE');
  writeln(lst,#12,'DECISION   SUPPORT   SYSTEM':53);
  writeln(lst,conststr('-',47):63);
  writeln(lst);
  writeln(lst,'COST  EFFECTIVENESS  ANALYSIS':54);
  writeln(lst,'FOR':41);
  writeln(lst,'CONTROL  &  SECURITY  OF  COMPUTER
                                      SYSTEMS.':62);
  writeln(lst);
  writeln(lst);
```

```
writeln(lst,'EXPECTED LOSSES CAUSED BY EXPOSURES FOR WORK'
                                          +' '+cproblem:66);
i := 9;
writeln(lst);
writeln(lst);
writeln(lst);
openfile(file1,dr+cproblem+'.dxp',sizeof(expsr));
openindex(index1,dr+cproblem+'.ixp',
                          sizeof(expsr.index),0);
writeln(lst,'THE WEIGHTED METHOD':48);
writeln(lst,conststr('=',80));
writeln(lst,'POTENTIAL ERRORS':37,'AMOUNT OF':28,
                                          'PROB/TY OF':15);
writeln(lst,'DAMAGE':64,'OCCURENCE':15);
writeln(lst,conststr('-',80));
clearkey(index1);
j := 17;
with  expsr  do
repeat
  nextkey(index1,rn,idx);
 if  ok  then
  begin
    j := j + 1;
    getrec(file1,rn,expsr);
    k := 50 - length(description);
    writeln(lst,index,'    ',description,conststr('  ',k),
                          damage:11,probability:12)
  end
until  not ok;
writeln(lst,conststr('=',80));
i := j - i + 2;
if  j + i > 56  then
  write(lst,#12);
writeln(lst);
writeln(lst);
writeln(lst);
writeln(lst,'THE P.E.R.T  METHOD':48);
writeln(lst,conststr('=',80));
writeln(lst,'POTENTIAL  ERRORS':37,'AMOUNT OF DAMAGE':37);
writeln(lst,'smallest':61,'m.likely':10,'largest':9);
writeln(lst,conststr('-',80));
clearkey(index1);
```

```
  with  expsr  do
  repeat
    nextkey(index1,rn,idx);
    if  ok  then
    begin
      getrec(file1,rn,expsr);
      k := 50 - length(description);
      writeln(lst,index,'    ',description,conststr('   ',k),
                        smallest:8,mostlikely:10,largest:9)
    end
  until  not ok;
  writeln(lst,conststr('=',80));
  writeln(lst);
  writeln(lst);
  writeln(lst);
  writeln(lst,'THE RANKING  METHOD':48);
  writeln(lst,conststr('=',80));
  writeln(lst,'POTENTIAL ERRORS':37,
                            'ESTIMATION OF PROBABILITY':42);
  writeln(lst,'OF OCCURENCE AND DAMAGE':78);
  writeln(lst,'Rank P':65,'Rank Q':11);
  writeln(lst,conststr('-',80));
  clearkey(index1);
  with  expsr  do
  repeat
    nextkey(index1,rn,idx);
    if  ok  then
    begin
      getrec(file1,rn,expsr);
      k := 50 - length(description);
      writeln(lst,index,'    ',description,conststr('   ',k),
                                        rankP:11,rankQ:11)
    end
  until  not ok;
  writeln(lst,conststr('=',80));
  closefile(file1);
  closeindex(index1)
end;
```

```
overlay procedure  printsetfile;
label
  cancel;
var
  method    : char;
  i,j,k,rn  : integer;
  mthd      : string[17];
  header    : string[80];
begin
  action('PRINTER / SET FILE');
  fillchar(mthd,sizeof(mthd),0);
  fillchar(header,sizeof(header),0);
  fillchar(comb,sizeof(comb),0);
  select('SELECT    W)eighted,  P).e.r.t  or  R)anking  : ',
                                  ['W','P','R'],method);

  case  method of
    'W' : begin
            mthd := 'WEIGHTED METHOD: ';
            comb := wcombindex
          end;
    'P' : begin
            mthd := 'P.E.R.T. METHOD: ';
            comb := pcombindex
          end;
    'R' : begin
            mthd := 'RANKING  METHOD: ';
            comb := rcombindex
          end
  end; {of case}
  openfile(file1,dr+cproblem+'.'+method+'dt',sizeof(st));
  if  not ok  then
  begin
    message('THERE IS NO FILE FOR THE '+mthd);
    goto cancel
  end
  else
    closefile(file1);
  write(lst,#12);
  writeln(lst);
  writeln(lst);
  writeln(lst,'D E C I S I O N   S U P P O R T
                                  S Y S T E M':62);
  writeln(lst,conststr('-',47):63);
  writeln(lst);
  writeln(lst,'COST  EFFECTIVENESS  ANALYSIS':54);
  writeln(lst,'FOR':41);
  writeln(lst,'CONTROL  &  SECURITY  OF  COMPUTER
                                      SYSTEMS.':62);
  writeln(lst);
```

```
writeln(lst);
writeln(lst,mthd+'CONTROL  SETS  FOR  WORK '+cproblem:62);
writeln(lst);
writeln(lst);
writeln(lst,'CONTROL  ACTIVITIES  USED  BY  THE  CONTROL
                                                SETS:');
openfile(file1,dr+cproblem+'.dcl',sizeof(ctrl));
openindex(index1,dr+cproblem+'.icl',sizeof(ctrl.index),0);
clearkey(index1);
j := 14;
for  i := 1 to maxctrl  do
if  comb[i] <> ''  then
begin
  findkey(index1,rn,comb[i]);
  if  ok  then
  begin
    getrec(file1,rn,ctrl);
    writeln(lst,ctrl.index,':          ',ctrl.description);
    j := j + 1
  end
end; {of for/if}
closefile(file1);
closeindex(index1);
writeln(lst);
writeln(lst);
header := '          CONTROL ACTIVITIES                '+
          '   VALUE '+'     COST '+'  EXP.COST '+'  BCR';
writeln(lst,header);
writeln(lst,conststr('-',80));
openfile(file1,dr+cproblem+'.'+method+'dt',sizeof(st));
openindex(index1,dr+cproblem+'.'+method+'ic',
                                    sizeof(st.Ck),1);
clearkey(index1);
j := j + 4;
k := 1;
with  st  do
repeat
  nextkey(index1,rn,idx);
  if  ok  then
  begin
    j := j + 1;
    if  j > 56  then
    begin
      j := 5;
      k := k + 1;
      write(lst,#12);
      write(lst,mthd+'CONTROL SETS FOR WORK '+
                                  cproblem:50);
      write(lst,'page ':24,k:2);
      writeln(lst);
```

```
        writeln(lst);
        writeln(lst);
        writeln(lst,header);
        writeln(lst,conststr('-',80))
      end;
      getrec(file1,rn,st);
      for  i := 1 to maxctrl  do
      if  setcomb[i] <> ''  then
        write(lst,setcomb[i]+',')
      else
        write(lst,'    ');
      write(lst,Vk:10,Ck:10,TCk:10,BCR:8);
      writeln(lst)
    end
  until  not ok;
  writeln(lst);
  closefile(file1);
  closeindex(index1);
  cancel:
end;

begin  {of printfiles}
  ans := ' ';
  while  ans <> '5'  do
  begin
    printmenu;
    select('SELECT 1,2,3,4 or 5 :',['1'..'5'],ans);
    if  (ans <> '1') and (ans <> '5')  then
    begin
      clearframe;
      message('TURN YOUR PRINTER ON.');
      wait
    end;
    case  ans  of
      '1' : help('P');
      '2' : exposuretable;
      '3' : controleffectable;
      '4' : printsetfile;
    end {of case}
  end; {of while}
end;
```

```
BEGIN   (OF SENSITIVITYANALYSIS)
  fillchar(key,sizeof(key),0);
  fillchar(idx,sizeof(idx),0);
  ans := ' ';
  while  ans <> '5'  do
  begin
      sensanalymenu;
      select('SELECT   1,2,3,4   or   5  :',['1'..'5'],ans);

    case  ans  of
      '1' : help('S');
      '2' : controlstrategy;
      '3' : graphics;
      '4' : printfiles;
    end (of case)

  end; (of while)
END;
```

```
procedure  help(ch : char);
{$I-}
var
  fl   : text;
  line : string[80];
  i,j  : integer;
begin
  assign(fl,'HELP'+ch+'.TXT');
  reset(fl);
  if  IOresult = 0  then
  begin
    clrscr;
    i := 0;
    while  not eof(fl)  do
    begin
      readln(fl,line);
      writeln(line);
      i := i + 1;
      if  i = 22  then
      begin
        wait;
        i := 0;
        clrscr
      end;
      if  eof(fl)  then
      wait
    end;
    clrscr;
    makeframe;
    putdate;
    problemfield(cproblem)
  end
end;
```

LIST OF REFERENCES

1. King, J. L. and Schrems, E. L., "Cost-Benefit Analysis in Information Systems Development and Operation", Computing Surveys, v. 10, no. 1, pp. 19-34, 1978.

2. Dowell, C. D. and Hall, J. A., "EDP Controls with Audit Cost Implications", Journal of Accounting, Auditing and Finance, pp. 30-40, 1981.

3. Martin, J., Security, Accuracy, and Privacy in Computer Systems, Prentice-Hall, p. 3, 1973.

4. Bui, T. X., A Cost-Effectiveness Analysis for Control and Security of Computer Systems, February 1985.

5. Bui, T. X. and Pasquier, J., Decision Support Systems: A Systemic Approach, p. 8, November 1983.

6. Bui, T. X. and Pasquier, J., Decision Support Systems: A Systemic Approach, p. 10, November 1983.

7. IBM Research Report RJ1382, A Case Study of Nonprogrammer Interactive Problem Solving, by E. D. Carlson and J. A. Sutton, April 1974.

8. Newell, A. and Simon, H. A., Human Problem Solving, Prentice-Hall, 1972.

9. McKenney, J. L. and Keen, P. G. W., "How Manager's Minds Work", Harvard Business Review, pp. 79-90, May-June 1974.

10. Kroenke, D. M., Database Processing: Fundamentals, Design, Implementation, Science Research Associates, p. 179, 1983.

11. Peterson, J. L. and Silberschatz, A., Operating System Concepts, Addison-Wesley, pp. 189-193, 1983.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defence Technical Information Center Cameron Station, Alexandria, Virginia 22304-6145	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93943-5100	2
3.	Department Chairman, Code 54 Department of Administrative Sciences Naval Postgraduate School Monterey, California 93943-5100	1
4.	Dr. Tung X. Bui, Code 54DH Department of Administrative Sciences Naval Postgraduate School Monterey, California 93943-5100	5
5.	LT E. A. Prevenas Patroklou 14 Moschato Athens, Greece	7
6.	Computer Technology Programs, Code 37 Naval Postgraduate School Monterey, California 93943-5100	1